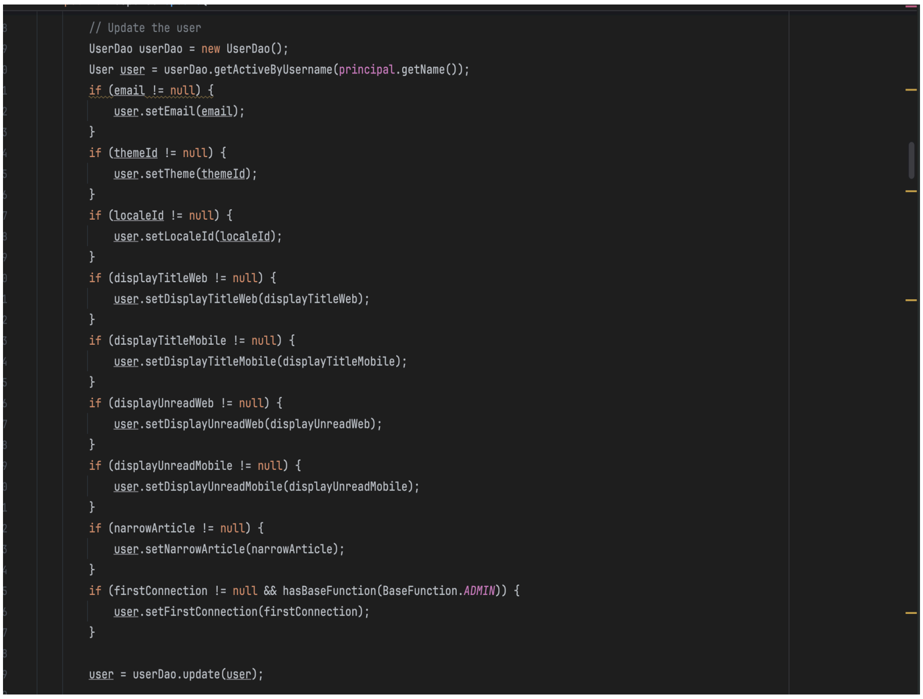
3C : Code Refactoring using LLMs

1.UserResource.java (Design smell : Cyclically Dependent Modularisation)

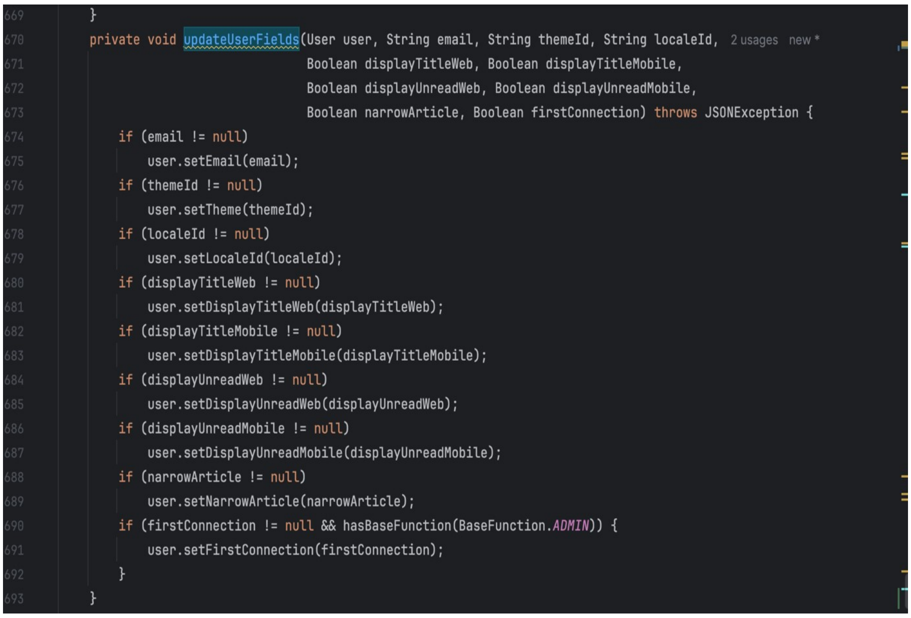
Original Code:



```
// Update the user
UserDao userDao = new UserDao();
User user = userDao.getActiveByUsername(principal.getName());
if (email != null) {
    user.setEmail(email);
}
if (themeId != null) {
    user.setTheme(themeId);
}
if (localeId != null) {
    user.setLocaleId(localeId);
}
if (displayTitleWeb != null) {
    user.setDisplayTitleWeb(displayTitleWeb);
}
if (displayTitleMobile != null) {
    user.setDisplayTitleMobile(displayTitleMobile);
}
if (displayUnreadWeb != null) {
    user.setDisplayUnreadWeb(displayUnreadWeb);
}
if (displayUnreadMobile != null) {
    user.setDisplayUnreadMobile(displayUnreadMobile);
}
if (narrowArticle != null) {
    user.setNarrowArticle(narrowArticle);
}
if (firstConnection != null && hasBaseFunction(BaseFunction.ADMIN)) {
    user.setFirstConnection(firstConnection);
}

user = userDao.update(user);
```

Apply Manual Refactoring : Created separate methode tom improve code readibility

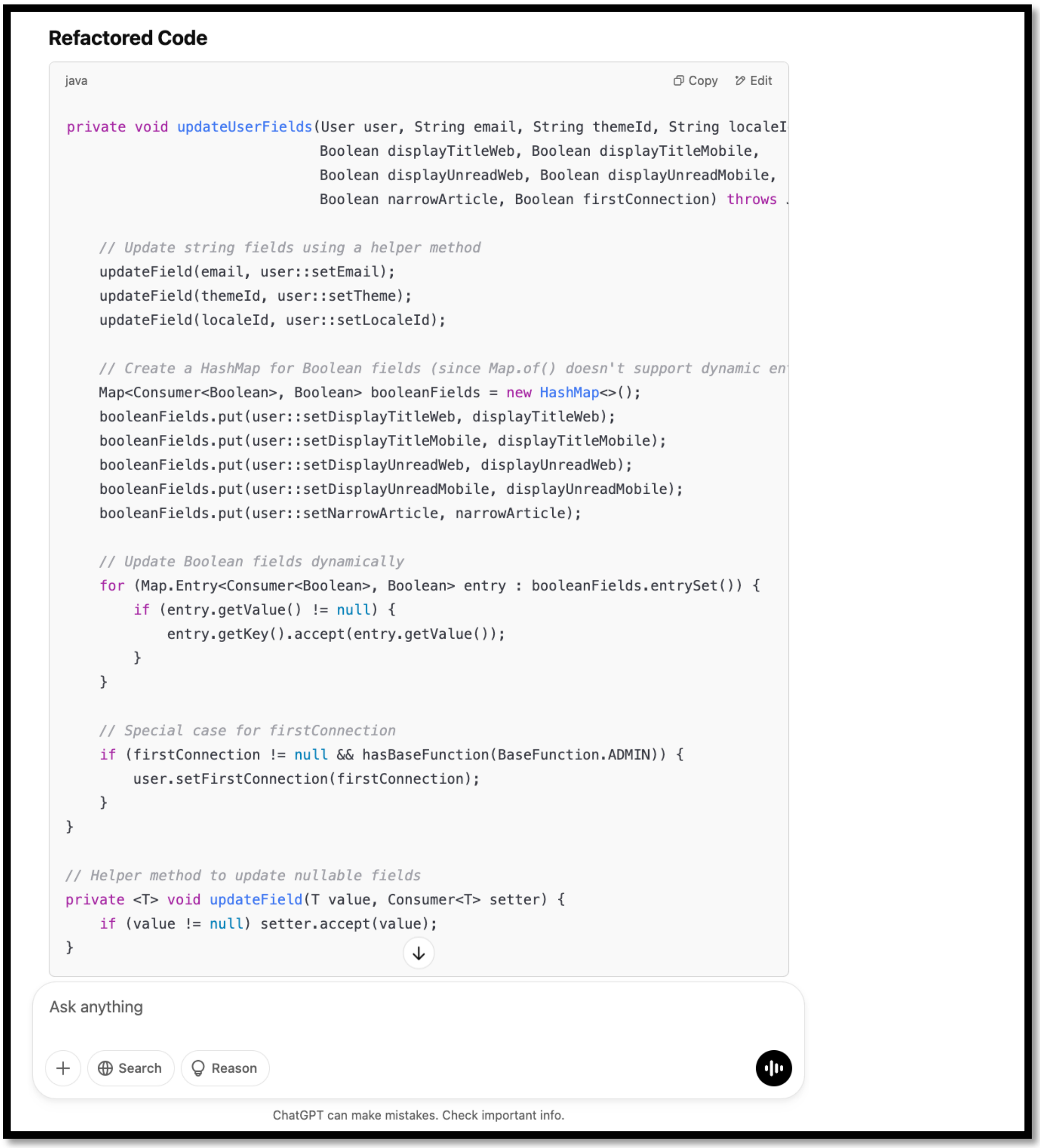


```
}
private void updateUserFields(User user, String email, String themeId, String localeId,
                              Boolean displayTitleWeb, Boolean displayTitleMobile,
                              Boolean displayUnreadWeb, Boolean displayUnreadMobile,
                              Boolean narrowArticle, Boolean firstConnection) throws JSONException {
    if (email != null)
        user.setEmail(email);
    if (themeId != null)
        user.setTheme(themeId);
    if (localeId != null)
        user.setLocaleId(localeId);
    if (displayTitleWeb != null)
        user.setDisplayTitleWeb(displayTitleWeb);
    if (displayTitleMobile != null)
        user.setDisplayTitleMobile(displayTitleMobile);
    if (displayUnreadWeb != null)
        user.setDisplayUnreadWeb(displayUnreadWeb);
    if (displayUnreadMobile != null)
        user.setDisplayUnreadMobile(displayUnreadMobile);
    if (narrowArticle != null)
        user.setNarrowArticle(narrowArticle);
    if (firstConnection != null && hasBaseFunction(BaseFunction.ADMIN)) {
        user.setFirstConnection(firstConnection);
    }
}
```

Key Improvements to reduce cyclomatic complexity and **improve readability** by:

- **Using a Map** to store the field values dynamically.
- **Iterating over the Map** instead of multiple `if` conditions.

LLM Refactoring



Refactored Code

```java
private void updateUserFields(User user, String email, String themeId, String localeI
                              Boolean displayTitleWeb, Boolean displayTitleMobile,
                              Boolean displayUnreadWeb, Boolean displayUnreadMobile,
                              Boolean narrowArticle, Boolean firstConnection) throws .

    // Update string fields using a helper method
    updateField(email, user::setEmail);
    updateField(themeId, user::setTheme);
    updateField(localeId, user::setLocaleId);

    // Create a HashMap for Boolean fields (since Map.of() doesn't support dynamic en
    Map<Consumer<Boolean>, Boolean> booleanFields = new HashMap<>();
    booleanFields.put(user::setDisplayTitleWeb, displayTitleWeb);
    booleanFields.put(user::setDisplayTitleMobile, displayTitleMobile);
    booleanFields.put(user::setDisplayUnreadWeb, displayUnreadWeb);
    booleanFields.put(user::setDisplayUnreadMobile, displayUnreadMobile);
    booleanFields.put(user::setNarrowArticle, narrowArticle);

    // Update Boolean fields dynamically
    for (Map.Entry<Consumer<Boolean>, Boolean> entry : booleanFields.entrySet()) {
        if (entry.getValue() != null) {
            entry.getKey().accept(entry.getValue());
        }
    }

    // Special case for firstConnection
    if (firstConnection != null && hasBaseFunction(BaseFunction.ADMIN)) {
        user.setFirstConnection(firstConnection);
    }
}

// Helper method to update nullable fields
private <T> void updateField(T value, Consumer<T> setter) {
    if (value != null) setter.accept(value);
}
```

- **Fixes `Map.of()` limitation** → Uses `HashMap` for dynamic field updates.
- **Keeps the logic clean and scalable** → No repetitive `if` conditions.
- **Reduces cyclomatic complexity** → Uses `for-each` instead of multiple `if` statements.
- **Ensures null safety** → Only updates fields if the value is not `null`