Escalona, J.M.

Group 10

**Case Study 1 - Authentication**

|  |  |  |  |
| --- | --- | --- | --- |
| **Security Issue** | **Java File** | **Vulnerabilities** | **Solution** |
| No user verification | Login.java  SQLite.java | Users are not necessarily verified to exist on the database before proceeding to checking authenticity.  \*This is not strictly for security purposes but rather includes performance ones. | Login.java: (**IMPLEMENTED**)  Add user verification which will dictate whether the user authenticity will proceed or not. Assumes login input verification returns true.  SQLite.java: (**IMPLEMENTED**)  Implement function that would only return a Boolean value to indicate whether user exists or not instead of the already implemented SQLite.getUsers() which returns all users including plaintext passwords of users. |
| No input validation on register input fields | Register.java | Inputs for user account credentials do not require minimum credential requirements such as minimum length (especially for passwords) and character composition.  Action of register was also found to be lost and accessible (public) on another class file Frame.java | Register.java: (**IMPLEMENTED**)   1. Add input validations for each user field such that it meets the following:    1. Uppercase letters (Username only): A-Z    2. Lowercase letters: a-z    3. Numbers: 0-9    4. Symbols (Username): \_-.    5. Symbols (Password): ~`!@#$%^&\*()\_-+={[}]|\:;"'<,>.?/ 2. Add username availability checking. |
|  |  |  |  |