COMP 730 + 830 Final Software Project Update 2

**Team Members:**

Charishma Bandari and Ben Heney

**Past Project Progress:**

Created the User.java class to handle user information and authentication features.

Implemented attributes such as first name, last name, date of birth, address, phone number, library card number, issue type, email, and zip code. Developed methods for registering a new user, updating user information, and retrieving user details.

Designed the BookCatalog.java class to manage the catalog of books in the system.

Implemented functionalities for adding new books, editing existing book entries, deleting books, and retrieving book details. Integrated the book catalog management system with the overall project structure. Ensured proper code structure and organization within the classes for readability and maintainability.

Utilized appropriate getters and setters for accessing and modifying class attributes.

Implemented necessary constructors and methods to handle user interactions and data operations.

**Past Team Member Activities:**

***Charishma Bandari:***

Responsible for creating the User.java class and implementing user authentication features. Developed methods for user registration, login, password management, and session handling. Integrated user authentication functionalities with the database and ensured data security. Conducted testing and debugging of backend functionalities, addressing issues promptly.

***Ben Heney:***

Ben has been developing and implementing the AppSettings and UserSettings classes, as well as the Database class and an installation script. This background infrastructure supports the main app in several ways:

1. AppSettings and install.sh compile the LibraryCatalog project with a unique encryption key that allows sensitive information to be stored locally. Specifically, database access credentials need to be retained in persistent memory so that users do not need to re-enter this information each time the application restarts. However, because the application is intended to run on public kiosks, it would be unwise to store this information without any kind of protection.

2. UserSettings stores information necessary to customize the application.

3. Database will provide an interface for, and a MySQL implementation of, storing and retrieving information from a remote database.

**To-Do Items** :

Our immediate focus is on advancing towards the 50% completion milestone. To achieve this, we have identified the following tasks:

Setting up a simple catalog browser for users to view book listings. Implement CRUD operations for book management, including adding new books, editing existing entries, and deleting books from the catalog. Create a database schema that includes tables for books, users, transactions (check-ins/outs), and any other relevant entities. Implement Java classes for interacting with the database, in one these classes implement methods for CRUD operations like addBook, updateBook, deleteBook, and getBookDetails to manage book entries in the database. Maintain a field in the Book table to track the availability status of each book, such as "Available," "Checked Out," or "On Hold." Update the availability status based on check-in/check-out transactions and user actions. Develop advanced search and filter functionalities to Implement advanced search and filter options allowing users to refine book searches based on criteria like author, genre, publication year, ISBN, and availability status. Provide filters to sort and categorize search results for easier navigation. Further enhance the check-in/check-out system by tracking book availability, due dates, and implementing notifications for overdue books and return reminders.

**Difficulties:**

Integrating the User.java and BookCatalog.java classes with other modules and ensuring seamless communication between them was challenging. Designing a scalable, maintainable, and extensible system architecture that aligns with project goals and requirements was difficult.

Designing a secure and convenient application has been a challenge. The current approach will expose the encryption key if the binary is decompiled. To eliminate this vulnerability, a password would be required on start-up. The password could then be hashed into an encryption key. This poses some issues for users as library staff would need to log in each time the application restarts. Furthermore, the password would need to be shared among library staff reducing its security. Lastly, implementing this would require more development resources be spent on features that do not contribute to the core functionality of the app. So, as a compromise, raw credentials are protected from being exposed through a local file system.