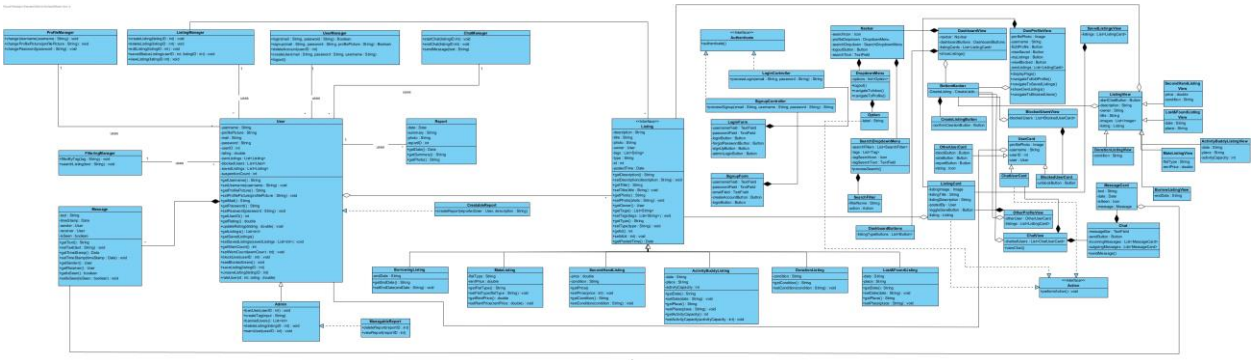
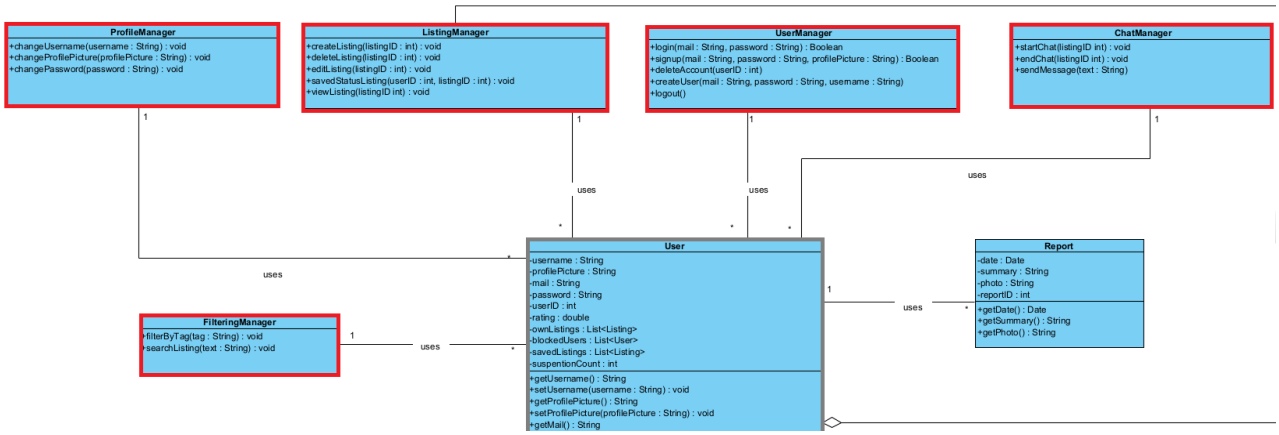
1. CLASS DIAGRAM



2. DESIGN PATTERNS

2.1. Singleton Pattern

While developing our web app, we chose to use singleton design pattern for our manager classes which are ProfileManager, ListingManager, FilteringManager, UserManager, and ChatManager. We made this decision to ensure that each of these manager classes has only one instance in throughout the application's lifecycle. Our aim was to have more control and management in our application and avoid unwanted creation of new objects. This decision also serves the purpose of having more united and consistent state in between the components of Student Stash. We will have better communication between different compounds of the application along with better overall performance. The singleton pattern, in this context, played a crucial role in a well-organized and scalable architecture, contributing to the overall robustness and maintainability of the application.



2.2. Strategy Pattern

In our web-application, we employed the strategy pattern within the listing interface, which encompasses the following listings; borrowing listing, mate listing, secondhand listing, activity-buddy listing, donation listing, and lost & found listing. So, when the users want to create different types of listings, the specific strategy class will be used for creation. With this design pattern, each listing type is defined independently, having a flexible and modular design. By encapsulating each listing type as a separate strategy class, we ensured that any changes or additions to the behavior of one type would not affect the others. It also supports code reuse, as

common functionalities can be implemented in a base strategy interface shared among all listing types. This design choice makes the system extensible without requiring modification to existing code making it scalable architecture for the application.