**Design Patterns**

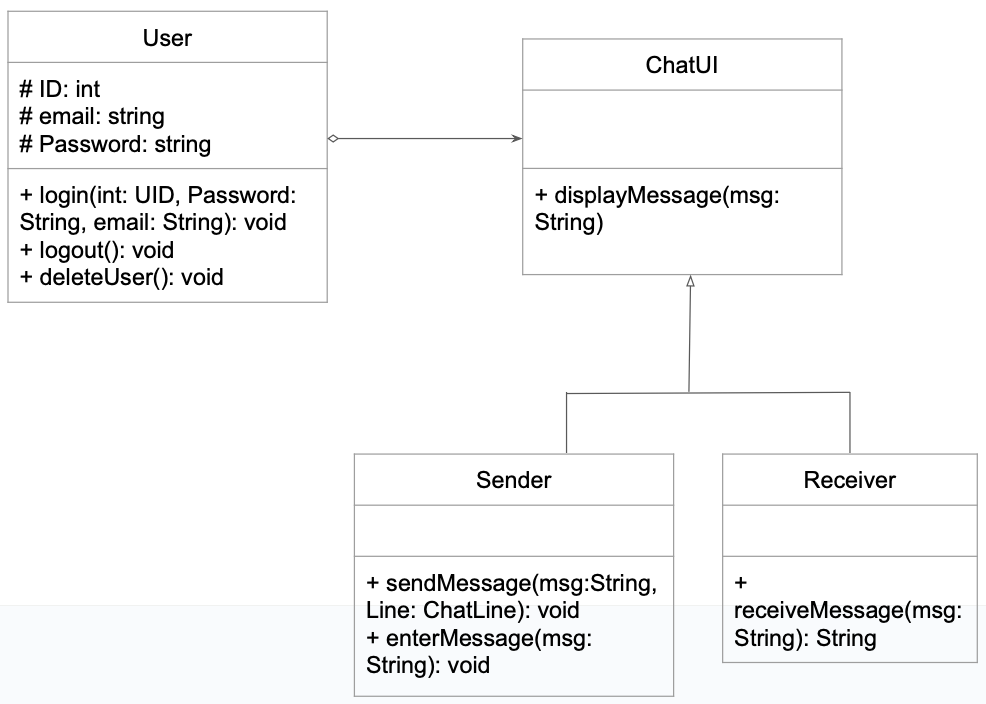
Four RoomUp, we will implement two different design patterns:

* Strategy

Strategy is a behavioral Design pattern that enables you to define a family of algorithms, put each of them into a separate class, and make their objects interchangeable.

With strategy, we capture the abstraction in an interface and bury implementation details in derived classes. Using strategy, there is no impact when the number of derived classes changes or the implementation of a derived class changes.

In RoomUp, Strategy would be the most suitable pattern to deal with the different types of users. Several options exist such as a user who has a room available, or someone who is seeking for a room, or a student looking for a roommate within a University system. Any of these users will use the same profile style, will be asked the same preferences, and will use the same UI.



* Template Method.

The template method is a behavioral Design pattern that would be used to share a common part of an implementation while allowing subclasses to refine other parts.

We would decide which steps of an algorithm are invariant and which are variant. Invariant parts would be defined in the abstract base class and variant steps would be given a default implementation.

In RoomUp, the template method would be a suitable way to deal with the different types of users. Whether the user is seeking for a room, advertising a room, or trying to find a college roommate, they all are subdivisions of the User class. Within the User, the login, logout, and deleteUser will be identical for each different subdivision. Additionally, the preferences option would vary from user type to user type. For example, a college user will have the same preferences as all other users and might have additional options such as on campus or off campus.

