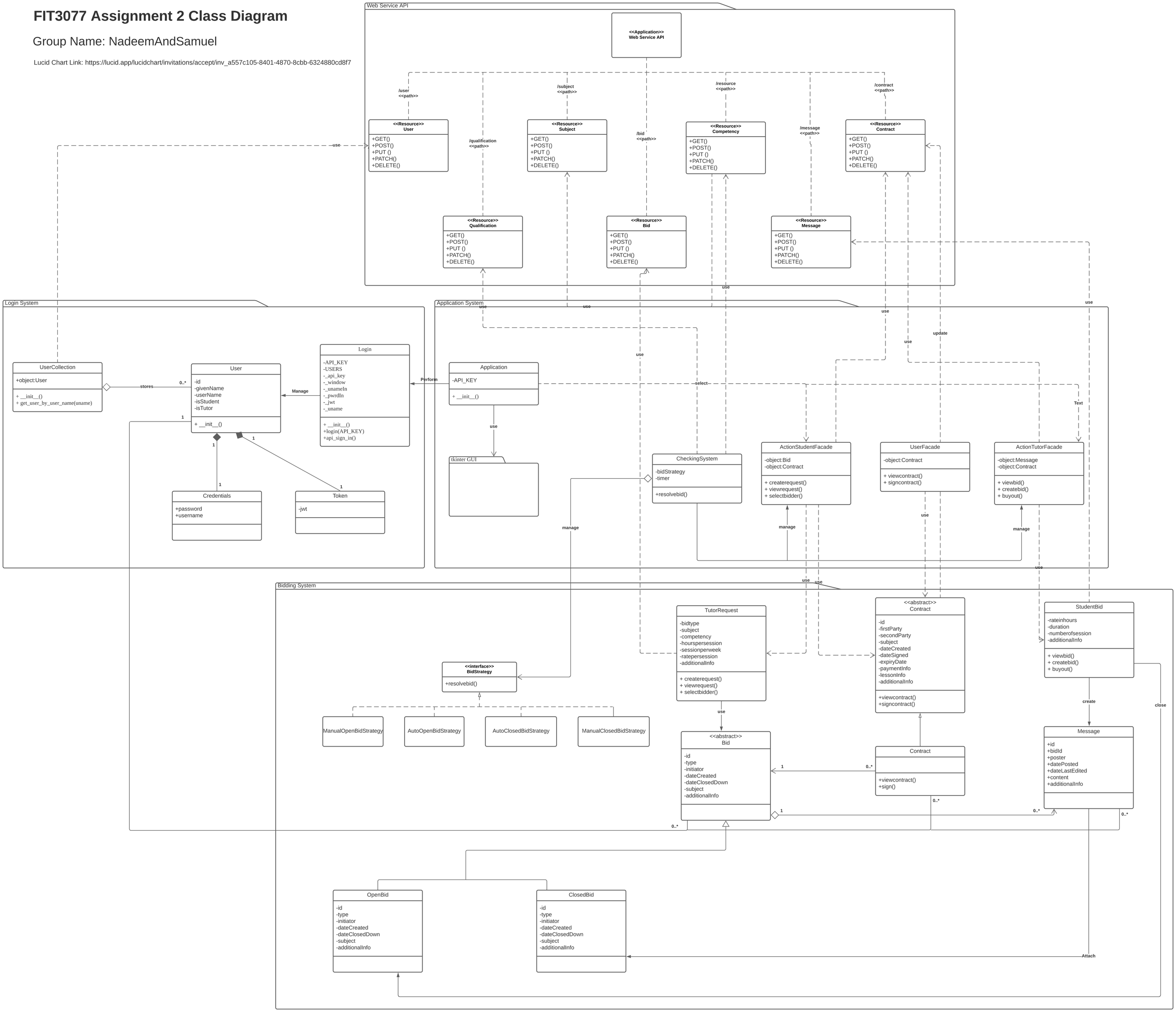
FIT3077 Assignment 2 Class Diagram

Group Name: NadeemAndSamuel

Lucid Chart Link: https://lucid.app/lucidchart/invitations/accept/inv_a557c105-8401-4870-8cbb-6324880cd8f7



FIT3077 Assignment 2 Design Rationale

Google Doc Link: https://docs.google.com/document/d/1u-StasBbIKkpKSfexs4929j36D9goCc9OKkBCcMq6qw/edit?usp=sharing

Group Name: NadeemAndSamuel

This report will outline the several design principles and design patterns adhered to during this project

User Class follows the **Singleton creational design pattern**. The reason why this design was used is to ensure that the system only has one instance of the user while providing a global access point to this instance.

UserCollection Class holds instances of User objects. The User Class is designed to adhere to the **Open Closed Principle (OCP)**. It is contained in a separate class to allow for an extension by other classes while remained closed for modification.

The **Facade structural design pattern** is used to provide a simple interface to a complex subsystem. In this system, we have many actions that clients can interact with. By providing only the relevant actions that the client is interested in, the implementation of the system is easier to be understood and maintained. Another advantage of using façade design is reducing coupling between multiple subsystems by requiring them to communicate only through facades.

The **Strategy behavioral design pattern** is used to handle the various ways how bids are resolved. For each type of bid, it can either be resolved automatically by the system timer or manually selected by the users. Each of these conditions is different and resolves each bid in a specific way. The advantage of this implementation is extracting all conditional algorithms into separate classes, all of which implement the same interface and allow it to switch from one algorithm solution to another during runtime.

The **Abstract Factory creational design pattern** is used to create similar but different types of bids. With this approach, the client code does not depend on concrete classes of bids but rather through their abstract interface. The benefit of abstraction is to support future implementation that the client may want to introduce in the future. For example, we can create a new concrete bid and contract class and modify its implementation slightly to accommodate new specifications.

The **Liskov Substitution Principle (LSP)** ensures that every subtype is expected to work when passed on to its superclass. We can observe that every bid of either open or closed type must be checked through a checking system for its appropriate level of competency and time left remaining. Its details are then sent to an interface for contract creation. The concrete bid child classes behavior created is consistent with the superclass.

Throughout the project, the classes were designed to adhere to the **Single Responsibility Principle (SRP)**, where each class represents a single entity and therefore has fewer reasons to change. This is shown for example in the Application class, where its only responsibility is instantiating the login class. The main goal achieved from adhering to this design principle (SRP) is to reduce complexity and to reduce the number of reasons a class has to change.

The overall design choices and implementation focus on the option of extending features for additional specifications in the future. The benefit of implementing it this way is that the design is more readable and easier to maintain.