**Object Oriented Programming Principles Used**

1. **Function overloading**

Overloaded constructors were used to handle different initialization formats for all the classes. The class diagram can be referred to for more information on the overloaded constructors

1. **Template Functions**

The *login* and *signup* functionality for the *Front Desk* and the *Librarian* were clubbed to make a generic template function. Doing this reduces coding complexity and also reduces the number of lines of code

1. **Operator Overloading**

The **insertion operator (<<)** was overloaded to print the date object when invoked by cout. For example, *cout<<d1* will print the *date, month and year* stored in the date object d1.

The **arithmetic (-) operator** was overloaded to calculate the difference between dates for return or for renewal of books. Printing *d1-d2* would return the difference between the dates d1 and d2 in terms of number of days

1. **Exception Handling**

Error handling was included to handle invalid data type inputs. An example for this is when the system expects an integer input but was fed with a character or a string data type.

1. **Friend classes and functions**

The *Front Desk Class* was made a friend to the *Customer Class* and *Book Class*. This was necessitated by the fact that the *Front-Desk Class* requires access to the private data of each book and every customer registered with the library.

1. **Data hiding and abstraction**

All the data pertaining to a class were kept in the private segment to facilitate data hiding.

Data abstraction was achieved by exposing only the function interfaces within the class definition and defining the class member functions of the class outside the class definition

1. **Inheritance**

The *Librarian Class, Customer Class* and *the Front Desk Class* were inherited from an abstract base *Person Class*. The *Person Class* stores data like *name* and *email ID* and supports the *print details* function which the three derived classes make use of