OOAD PROJECT FINAL REPORT (BUFF BEDS)

Team :

Gaurav Gandhi

Sarang Kulkarni

Akshit Shah

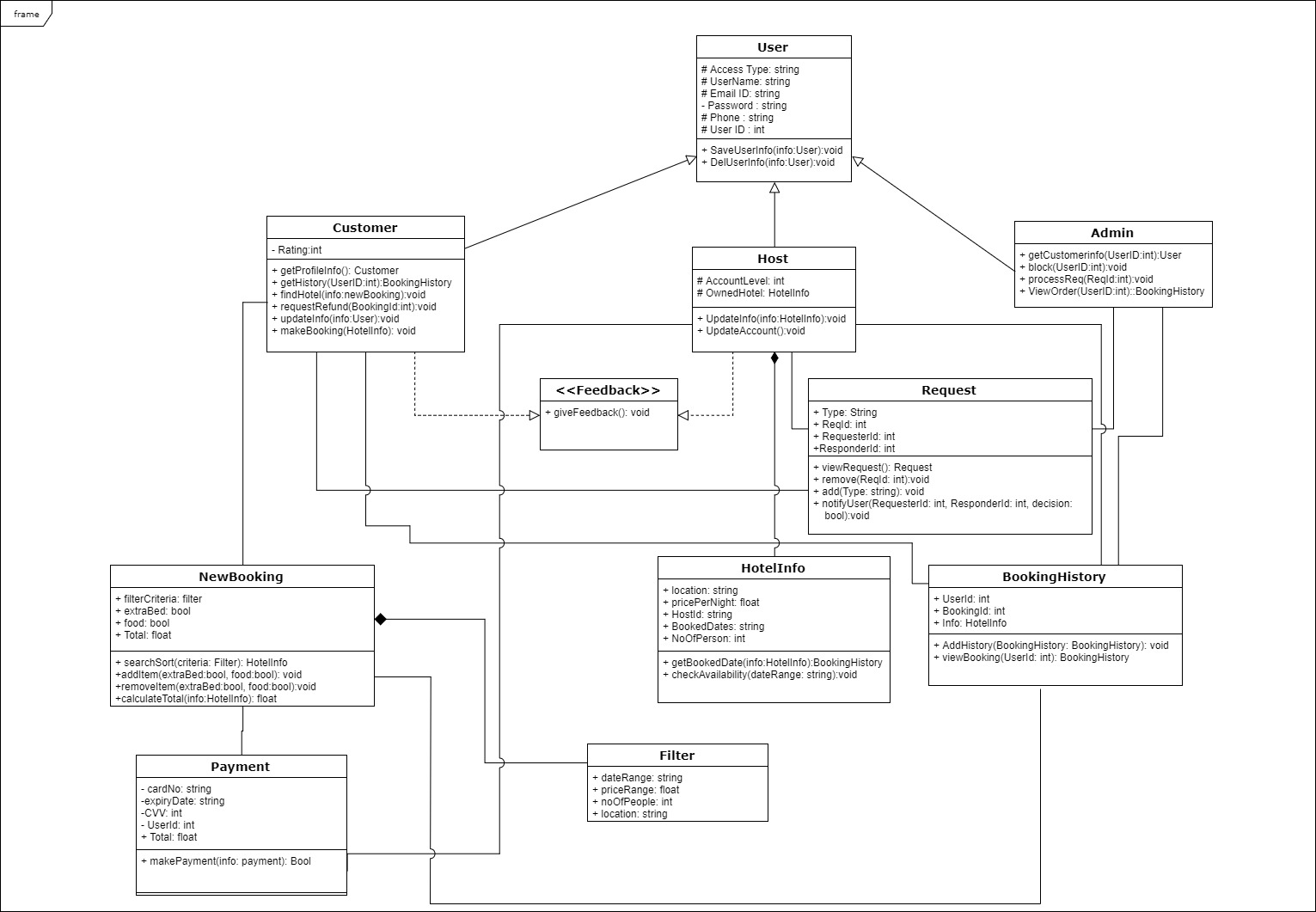
Title : Buff Beds

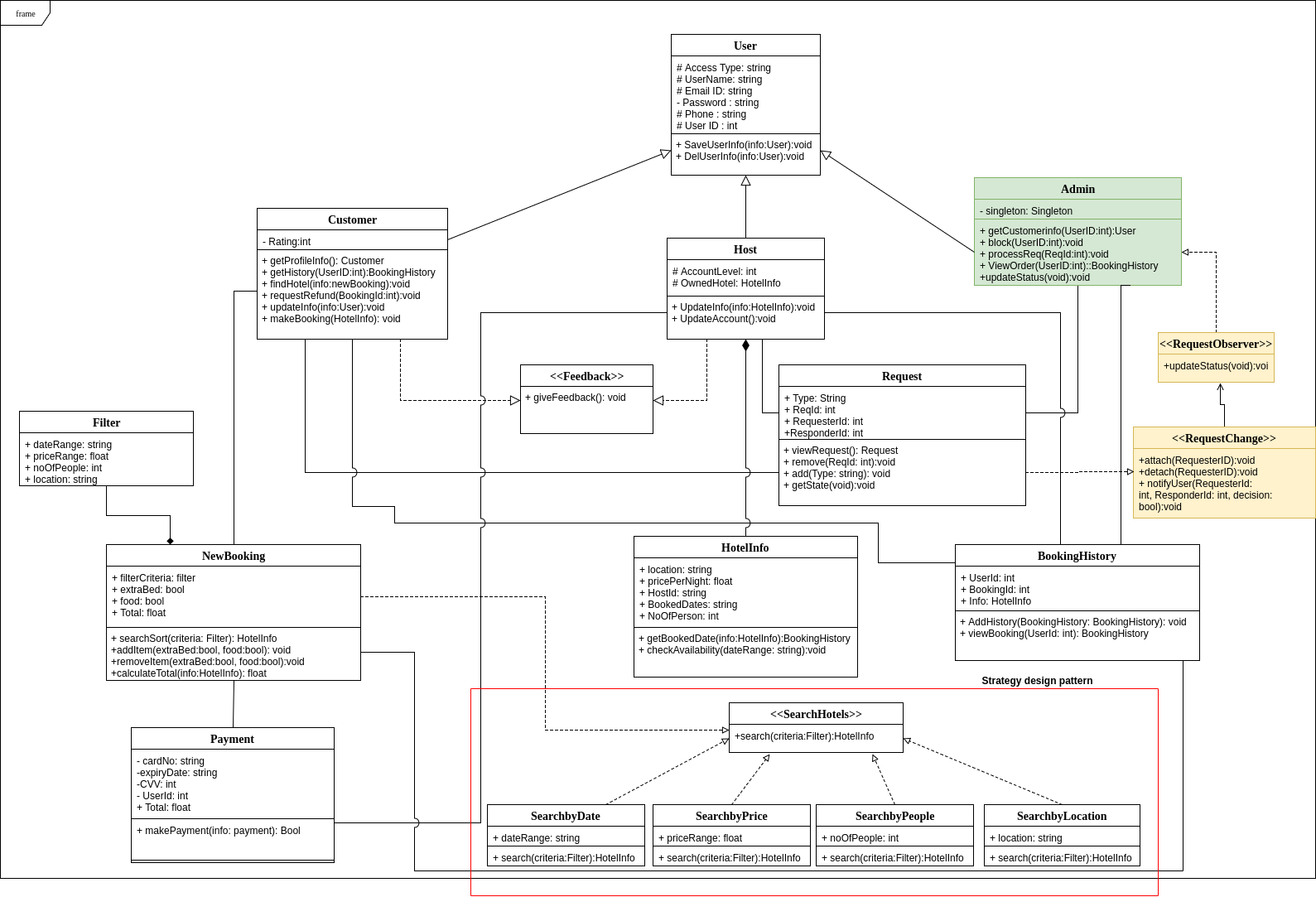
Description: A hotel/accommodation booking website, where End Users can have access to large number of houses and hotels for accommodation. Also house owners can post their houses for renting.

1. Implemented Features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Requirement** | **Topic Area** | **Actor** | **Priority** |
| UR-01 | Customer can sign up. | Login | Customer, Host | Critical |
| UR-02 | Host can Apply for hotel/SignUp | Login | Host, Admin | Critical |
| UR -03 | End Users can log in. | Login | Customer, Hosts,Admin | Critical |
| UR – 04 | Admin can block both hosts and Customers | Login | Admin | High |
| UR – 05 | Customer can search and sort hotels or rooms | Database | Customer | High |
| UR – 06 | Customer can checkout the hotel/room | Orders | Customer | High |
| UR – 07 | Customer cancel the booking | Orders | Customer | High |
| UR – 08 | Customer can request addons to their orders | Orders | Customer, Host | Low |
| UR – 09 | Hosts can upgrade their account | Database | Hosts | High |
| UR - 10 | Host  can update the info | Profile | Host | Medium |
| UR -11 | End User can rate and give feedbacks | Database | Customer, Host | Medium |
| UR – 12 | Admin can approve request for new hosts/request for update information | Database | Admin | High |
| UR - 13 | Admin can approve a refund | Orders | Admin | Medium |

1. Features which were not implemented

3. Original Class Diagram

Final Class Diagram 

Peach Color - Observer Pattern

Green Color – Singleton Pattern

Lot of changes were done to the design. Most important change was the implementation of the design patterns such as observer, singleton and strategy. Adding these design patterns made the implementation of some use cases simpler.

The small changes were addition of variables and

functions that became apparent while trying to implement the program. Planning the

design upfront was also very helpful for organized thinking which is required for

software design. We understood how the user requirements will be implemented in

software beforehand and this saved us from the constant struggle of changing our

implementation again and again when new user requirements came up. Also, with the

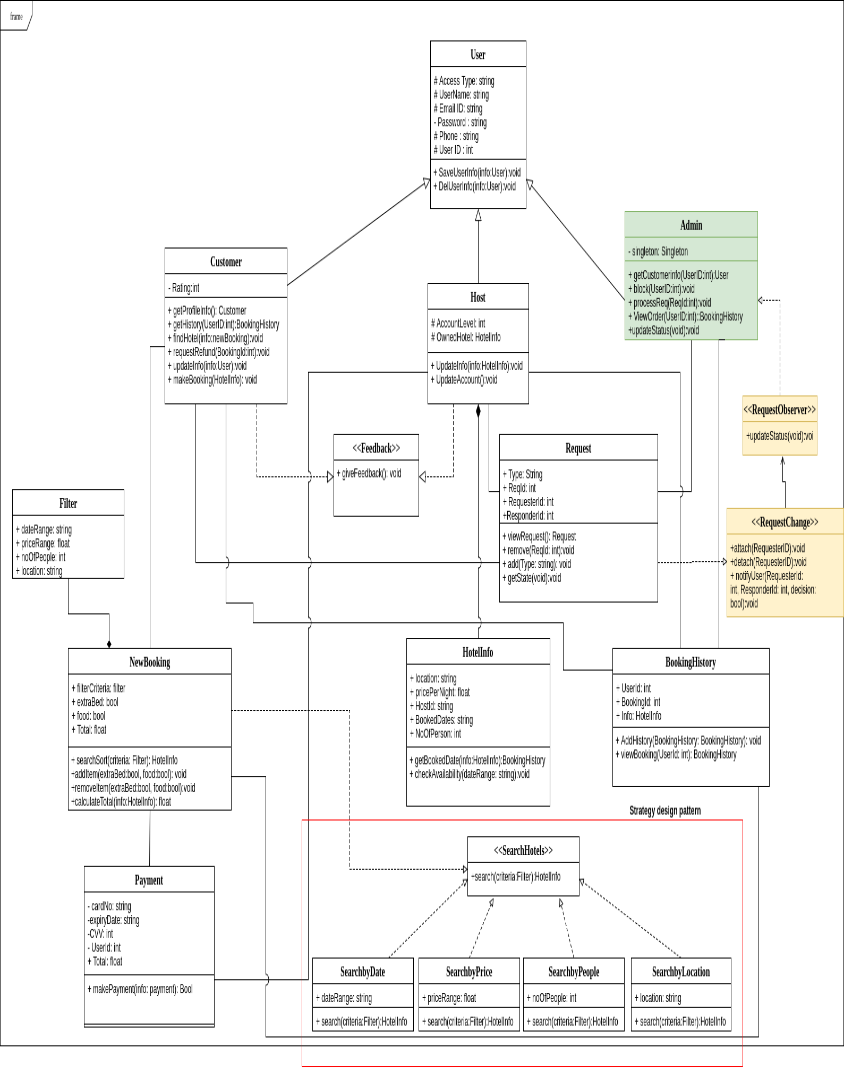
help of UML diagrams every team member could convey their ideas in a common

representational form. We gained knowledge about the actual form of Design Patterns

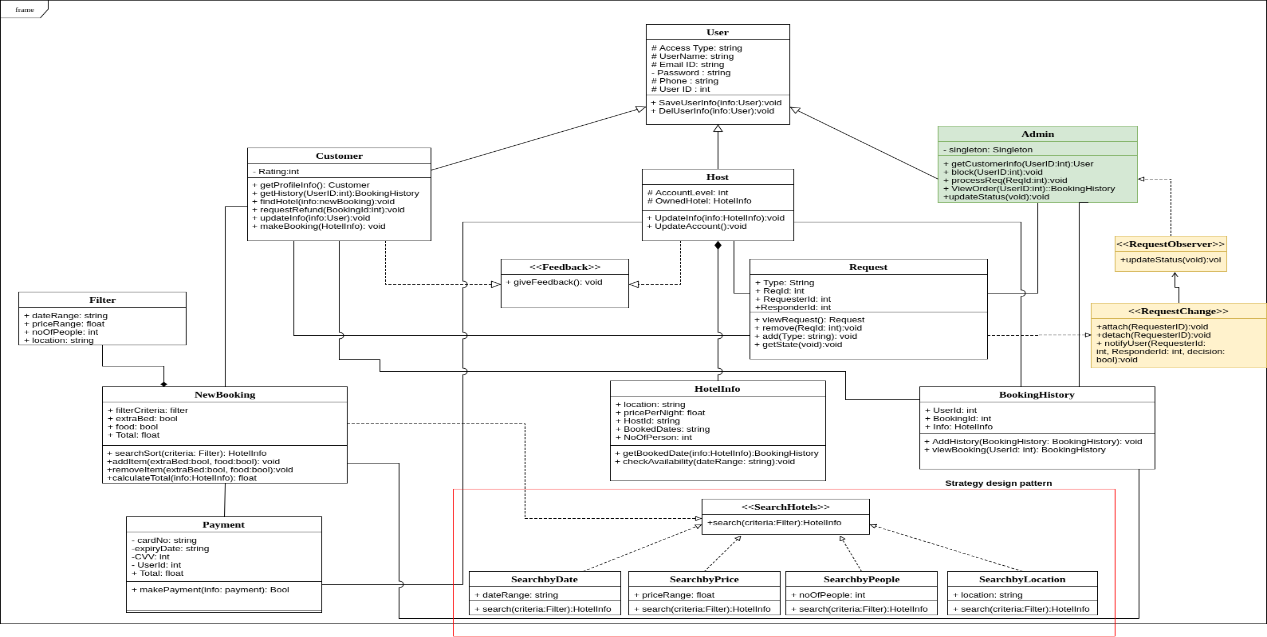
through the lectures and applied them during refactoring.

4.

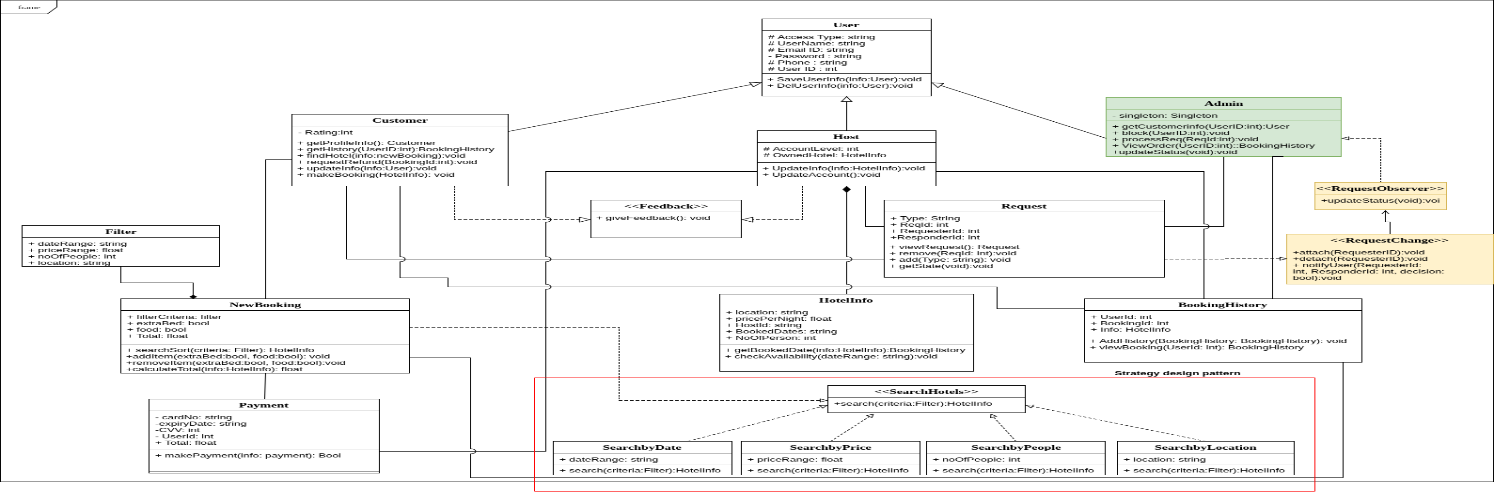
The observer design pattern is applied to the request use case which is used by the host to request changes to his/her profile. This is done because before implementing this the host had no way to get to know the status of his request. Through this change the host will be able to get the status of his/her request.



Singleton pattern: The admin class is made Singleton as for security reasons there should only be a single administrator account. Before this change it would have been possible to create many admin accounts.



Strategy pattern: While searching for a hotel, a user can search based on Date, Price, Number of people allowed or Location. Depending on the type of search, different algorithm would be used. For this purpose, the strategy design pattern has been used. When searching for a hotel, the interface Search is used, which has four realizations depending on the type of search, Search by Date, Search by Price, Search by People, and Search by Location.



5.

We have learned that it is extremely important to spend a good amount of time on

analysing and designing the system before implementing it. Software is naturally

intangible which makes it hard to convey meaning between use (the developers).

However, the diagrams that we created enabled us to share a common understanding

and made sure that we are all on the same page. The refactoring portion showed how

the original design is usually flawed and many iterations are necessary to ensure a

good design. The class diagram made us think thoroughly about the classes that we will

have along with attributes and methods. So, most of the implementation part was

creating those same exact classes. The design patterns that we learned during class

helped us solve some design issues in our system and will surely help use with future

design problems.