HostelChai.com

Contents:

USE CASE DIAGRAM

Use Case Diagram:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.
- A use case diagram should be simple and contains only a few shapes.

Use Case:

According to Wikipedia, In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal. The actor can be a human or other external system.

Actor:

- Actors may represent roles played by human users, external hardware, or other systems. Actors do not necessarily represent specific physical entities but merely particular facets of some entities that are relevant to the specification of its associated use cases.
- An Actor models a type of role played by an entity that interacts with the subject, but which is external to the subject.

How to Identify Actor:

The following questions can help to identify the actors of a system (Schneider and Winters - 1998):

- Who uses the system?
- Who installs the system?
- Who starts up the system?
- Who maintains the system?
- Who shuts down the system?
- What other systems use this system?
- Who gets information from this system?
- Who provides information to the system?
- Does anything happen automatically at a present time?

System Boundary:

- A system boundary is a rectangle that you can draw in a use-case diagram to separate the use cases that are internal to a system from the actors that are external to the system. A system boundary is an optional visual aid in the diagram; it does not add semantic value to the model.

Association:

- An association is the relationship between an actor and a business use case. It indicates that an actor can use a certain functionality of the business system.

- Unfortunately, the association does not give any information about the way in which the functionality is used. If a business use case includes several actors, it is not apparent in the use case diagram if each actor can conduct the business use case alone, or if the actors conduct the business use case together. In fact, association only means that an actor is involved in the business use case.

References:

- "OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, pp. 586–588". Archived from the original on 2010-09-23. Retrieved November 7, 2010.
- "Problems and Deficiencies of UML as a Requirements Specification, s.3.2" (PDF). Archived (PDF) from the original on 17 October 2010. Retrieved November 7, 2010.
- "UML 2 Specification". Retrieved July 4, 2012.
- System Analysis and Design 5th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth

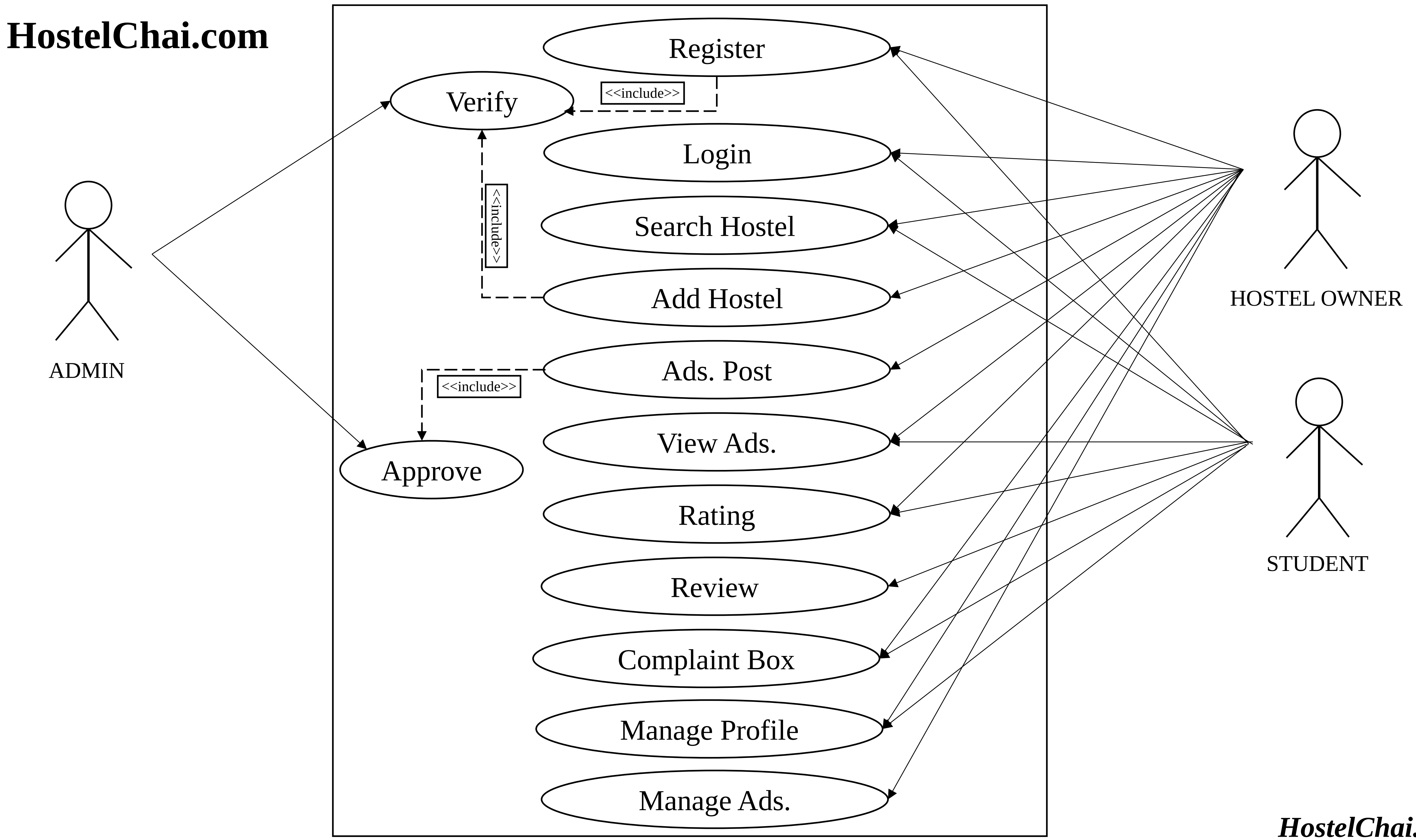
Actors of our Project:

- STUDENT
- HOSTEL OWNER
- ADMIN

Use Cases of our Project:

- Register
- Login
- Search Hostel
- Add Hostel
- Ads. Posting
- View Ads.
- Rating
- Review
- Complaint Box
- Manage Profile
- Manage Ads.

HostelChai.com



CLASS DIAGRAM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

What is class?

Class are a blueprint or a set of instructions to build a specific type of object. It is a basic concept of Object-Oriented Programming.

How we can identify who are the candidates as a class?

To identify the candidates as a class it needs to be physical objects, or a group of objects that are tangible and devices with which the application interacts. examples- books, cars, pressure sensors etc.

What is attribute?

Attribute represents the structure or the information of a class. It can be identified by any of these three ways

- ➢ Examining class definition
- ➢ Studying requirements
- ➢ Applying domain knowledge

What is method?

A method is a procedure associated with a message and an object. One of the most important capabilities that a method provides is method overriding which allows the same name to be used for multiple different kinds of classes.

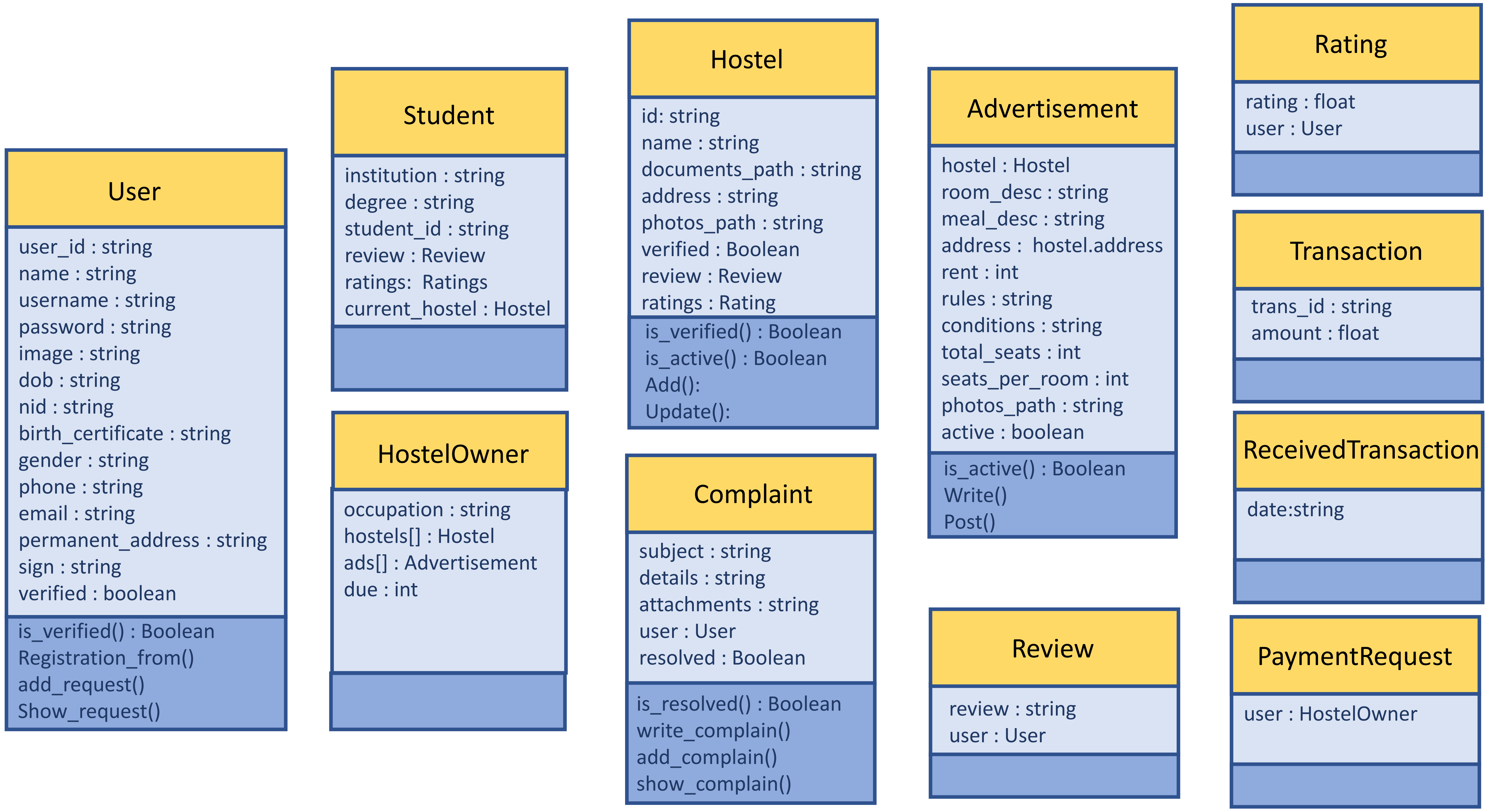
What is inheritance? And property of inheritance relationship

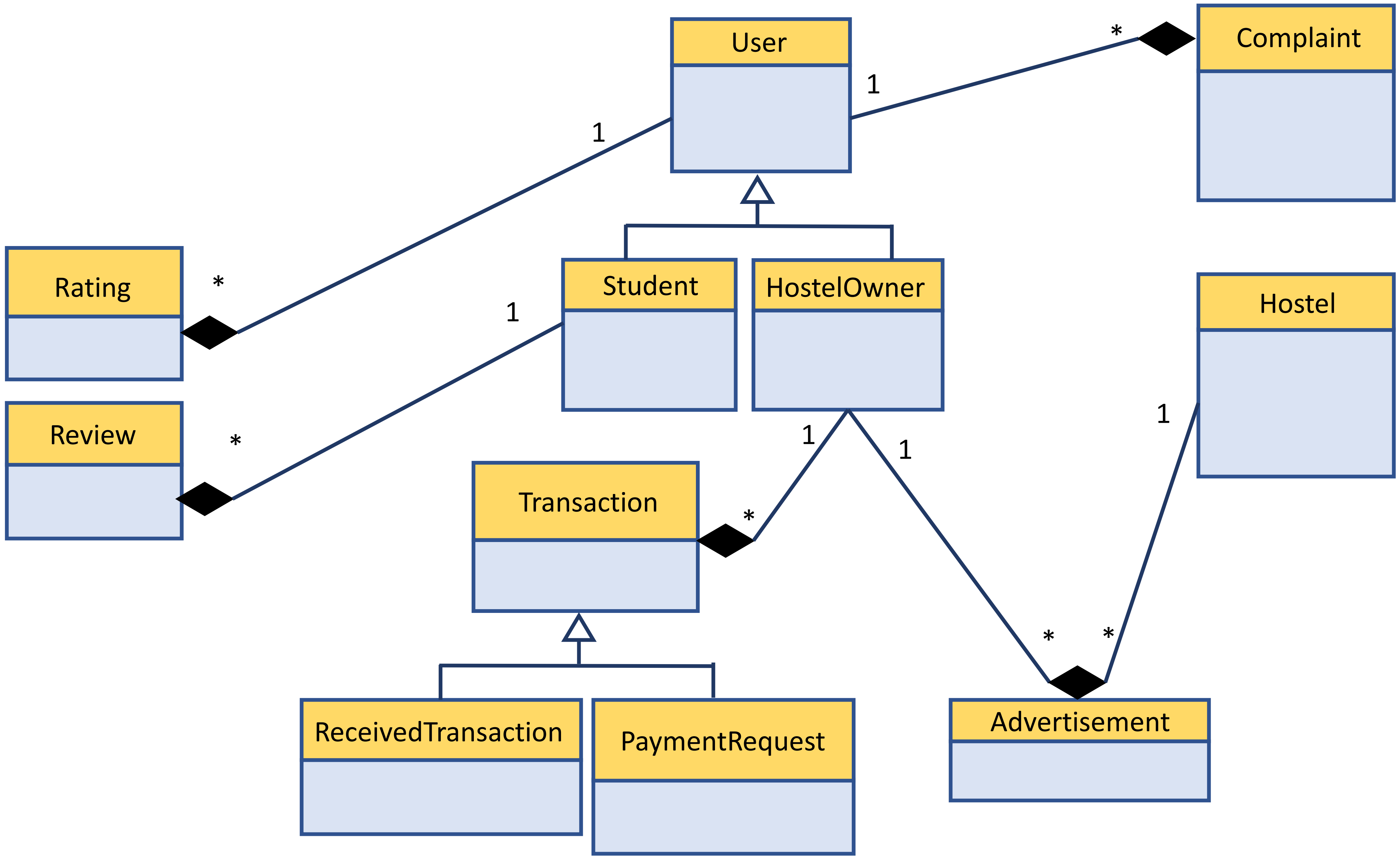
Inheritance is a mechanism in which one class acquires the property of another class. For example, a child inherits the traits of his/her parents.

With inheritance, we can reuse the fields and methods of the existing class. Hence, inheritance facilitates Reusability. From the domain view, an inheritance relationship organizes classes in hierarchical structures. This allows us to model a hierarchy of terms using generalizations and specializations.

- **References:**

- **https://www.guru99.com/java-oops-class-objects.html#:~:text=Class%20are%20a%20blueprint%20or,what%20the%20object%20will%20contain**
- **https://www.ibm.com/support/knowledgecenter/SSRTLW_9.6.1/com.ibm.xtools.viz.class.diagram.doc/topics/cattributes.html**
- System Analysis and Design 5th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth


User
user_id : string
name : string
username : string
password : string
image : string
dob : string
nid : string
birth_certificate : string
gender : string
phone : string
email : string
permanent_address : string
sign : string
verified : boolean
is_verified() : Boolean
Registration_from()
add_request()
Show_request()
Student
institution : string
degree : string
student_id : string
review : Review
ratings: Ratings
current_hostel : Hostel
HostelOwner
occupation : string
hostels[] : Hostel
ads[] : Advertisement
due : int
Hostel
id: string
name : string
documents_path : string
address : string
photos_path : string
verified : Boolean
review : Review
ratings : Rating
is_verified() : Boolean
is_active() : Boolean
Add():
Update():
Complaint
subject : string
details : string
attachments : string
user : User
resolved : Boolean
is_resolved() : Boolean
write_complain()
add_complain()
show_complain()
Advertisement
hostel : Hostel
room_desc : string
meal_desc : string
address : hostel.address
rent : int
rules : string
conditions : string
total_seats : int
seats_per_room : int
photos_path : string
active : boolean
is_active() : Boolean
Write()
Post()
Review
review : string
user : User
Rating
rating : float
user : User
Transaction
trans_id : string
amount : float
ReceivedTransaction
date:string
PaymentRequest
user : HostelOwner


User
Complaint
Rating
Student
HostelOwner
Hostel
Review
Transaction
ReceivedTransaction
PaymentRequest
Advertisement
1
*
1
*
1
*
1
*
1
*
1
*

SEQUENCE DIAGRAM

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

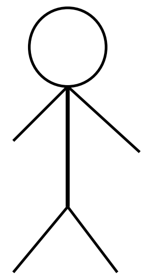
It shows the explicit sequence of messages that are passed between objects in a defined interaction. Since sequence diagrams emphasize the time-based ordering of the activity that takes place among a set of objects, they are very helpful for understanding real-time specifications and complex use cases.

Elements of a Sequence Diagram:

- Actor
- Object
- Lifeline
- Focus of Control
- Message
- Object Destruction

An actor:

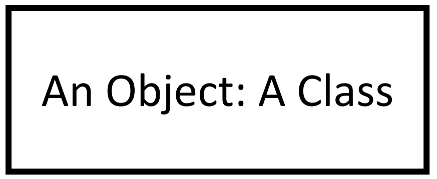
- ✓ Is a person or system that derives benefit from and is external to the system.
- ✓ Participates in a sequence by sending and/or receiving messages.
- ✓ Is placed across the top of the diagram.



An actor

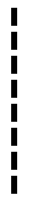
An object:

- ✓ Participates in a sequence by sending
- ✓ and/or receiving messages.
- ✓ Is placed across the top of the diagram.



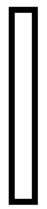
A lifeline:

- ✓ Denotes the life of an object during a sequence.
- ✓ Contains an X at the point at which the class no longer interacts.



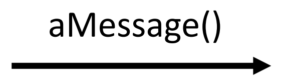
A focus of control:

Is a long narrow rectangle placed atop a lifeline.
Denotes when an object is sending or receiving messages.



A message:

- ✓ Conveys information from one object to another one.



Object destruction:

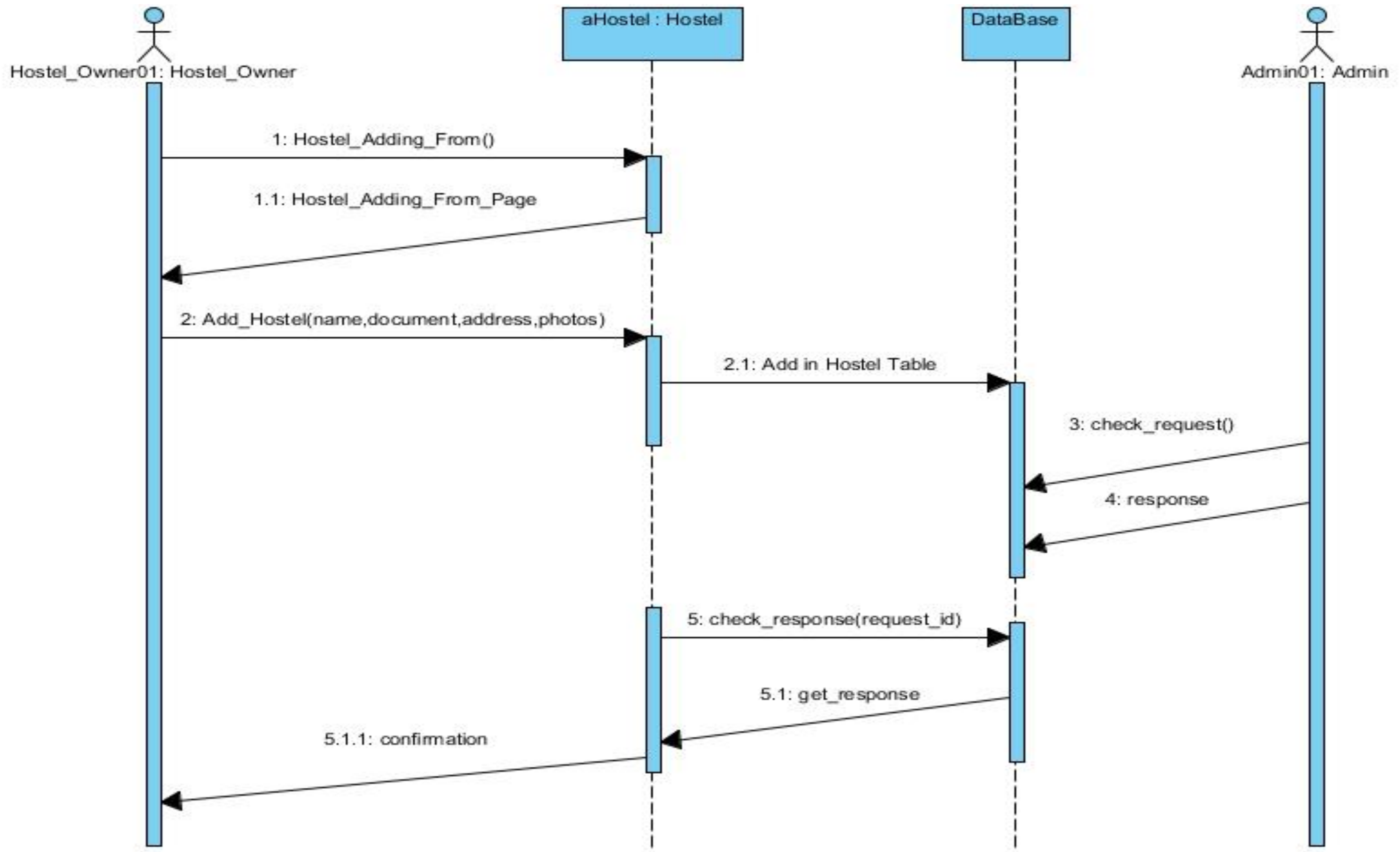
- ✓ An X is placed at the end of an object’s lifeline to show that it is going out of existence.

X

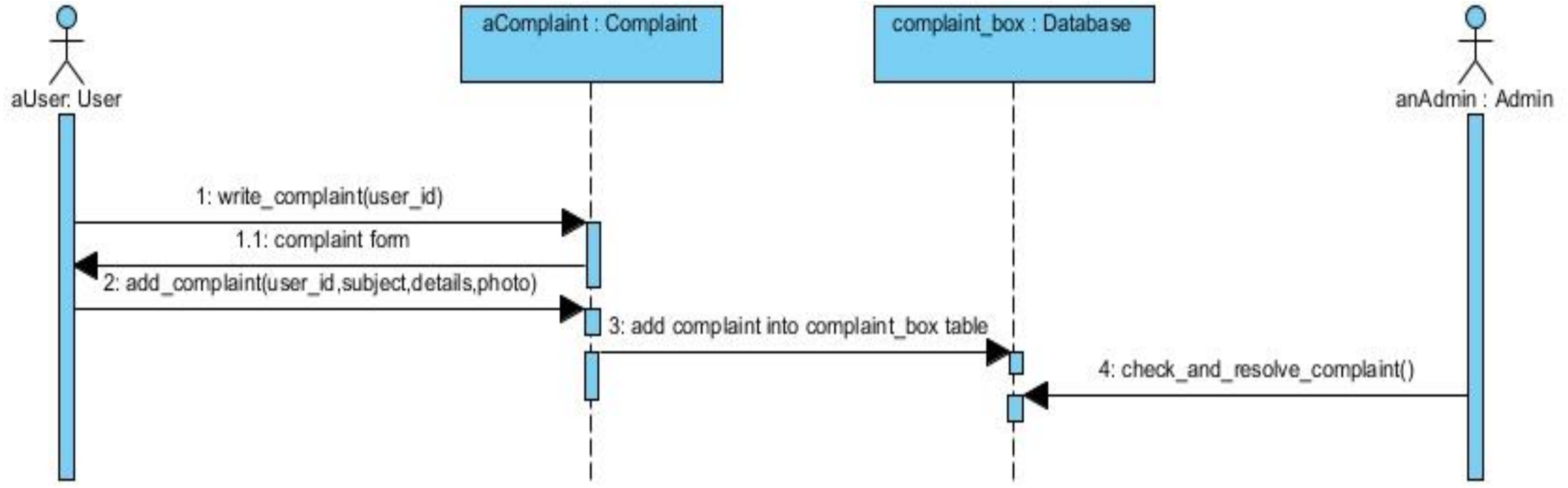
References:

- ❖ https://www.smartdraw.com/sequence-diagram/
- ❖ System Analysis and Design 5th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth

Use case 04: Add Hostel



Use case 09: Complaint Box


aUser: User
aComplaint : Complaint
complaint_box : Database
anAdmin : Admin
1: write_complaint(user_id)
1.1: complaint form
2: add_complaint(user_id,subject,details,photo)
3: add complaint into complaint_box table
4: check_and_resolve_complaint()

DEPLOYMENT DIAGRAM

What is a Deployment Diagram:

A UML deployment diagram is a diagram that shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams are a kind of structured diagrams used in modeling the physical aspects of a system or a platform. It shows how the end-users are accessing the system and how the system is serving the service. It only covers the basic structured view of the system without going into details.

Where and when to use Deployment diagram:

- When we need to show, with what the newly added system will interact or integrate with.
- How large or vast the system needs to be.
- Who/what will interact with the system.
- What kind of protocols does the system need to go through.
- What kind of HW/SW does the system need to function properly.
- Who/What can access which part and how'd they access it.

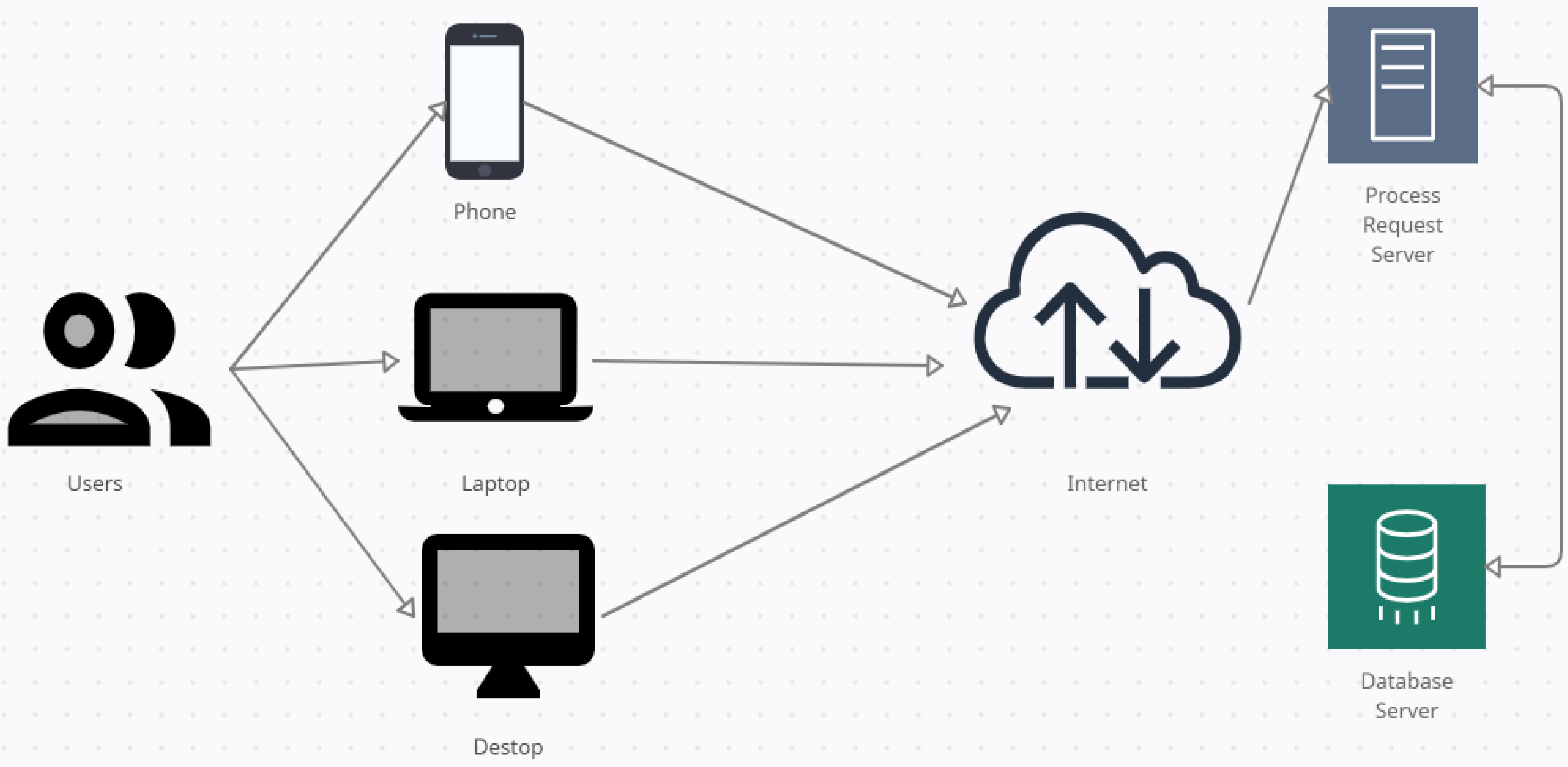
References:

- What is Deployment Diagram - VisualParadigm
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/
- Deployment Diagram – Wikipedia

https://en.wikipedia.org/wiki/Deployment_diagram

- Deployment Diagrams - CS dept of Loyla Marymount University

https://cs.lmu.edu/~ray/notes/deploymentdiagrams/


Phone
Process
Request
Server
Users
Laptop
Internet
Destop
Database
Server

STATE DIAGRAM

Specifically a state diagram describes the behavior of a single object in response to a series of events in a system. Sometimes it's also known as a Harel state chart or a state machine diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system.

The behavioral state machine diagram shows the different states that a single instance of a class passes through during its life in response to events, along with responses and actions.

State: A state is a set of values that describes an object at a specific point in time, and it represents a point in an object's life in which it satisfies some condition, performs some action, or waits for something to happen.

Event: An event is something that takes place at a certain point in time and changes a value(s) that describes an object, which in turn changes the object's state.

Basic Components of State Diagram:

- **Initial state –** We use a black filled circle represent the initial state of a System or a class.
- **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.
- **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.
- **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.

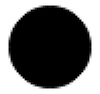


Figure – initial state notation

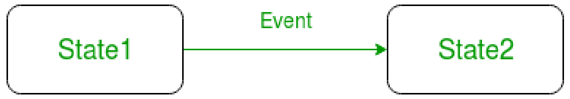


Figure – transition

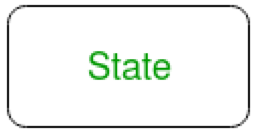


Figure – state notation

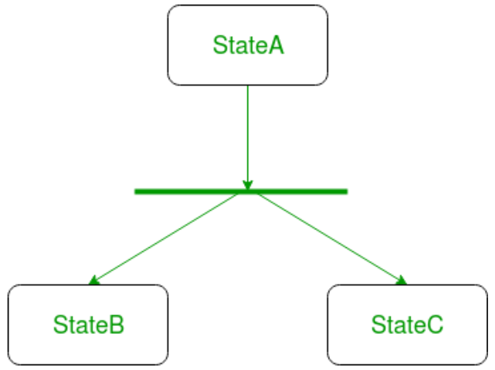


Figure – a diagram using the fork notation

Basic Components of State Diagram:

- **Join –** We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.

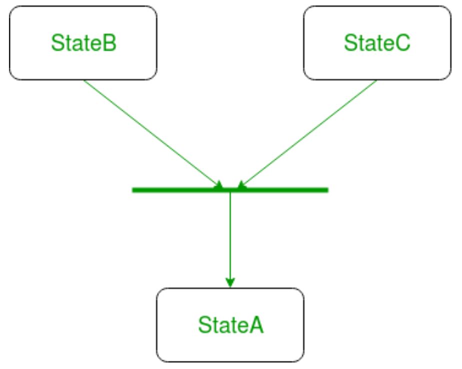


Figure – a diagram using join notation

- **Self transition –** We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.

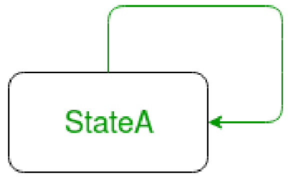


Figure – self transition notation

- **Composite state –** We use a rounded rectangle to represent a composite state also. We represent a state with internal activities using a composite state.

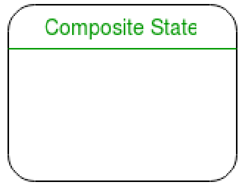


Figure – a state with internal activities

- **Final state –** We use a filled circle within a circle notation to represent the final state in a state machine diagram.

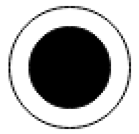


Figure – final state notation

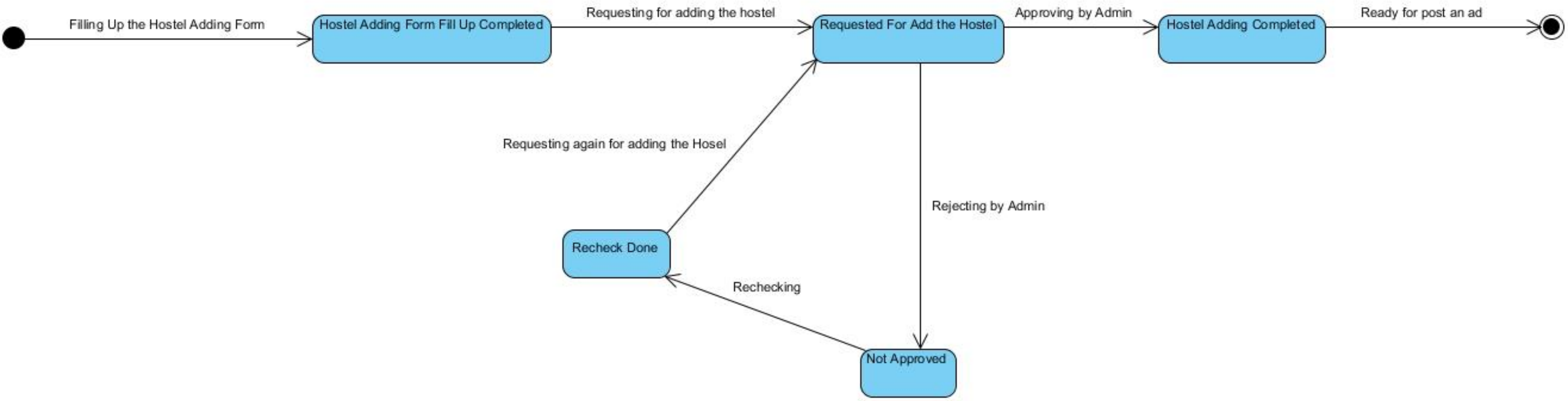
Steps to draw a state diagram –

1. Identify the initial state and the final terminating states.
2. Identify the possible states in which the object can exist (boundary values corresponding to different attributes guide us in identifying different states).
3. Label the events which trigger these transitions.

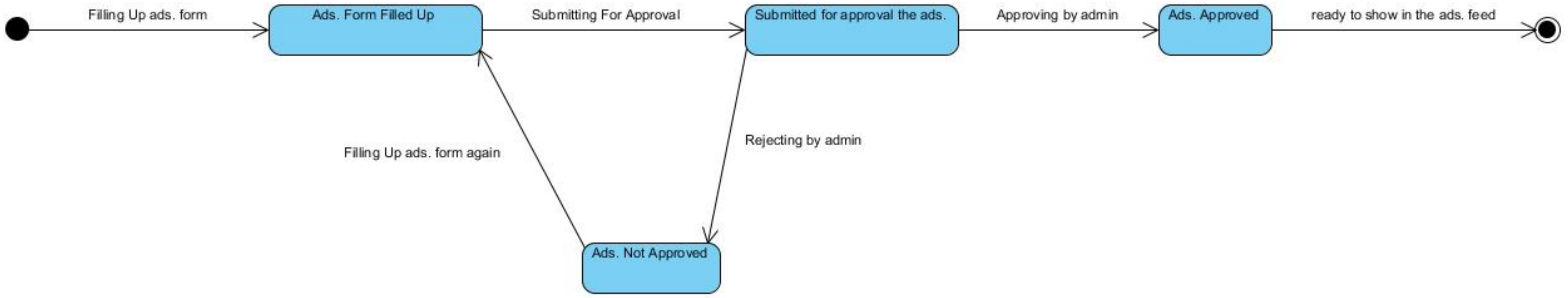
References:

- https://www.smartdraw.com/state-diagram/#:~:text=Specifically%20a%20state%20diagram%20describes,particular%20object%20within%20a%20system.
- System Analysis and Design 5th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth
- https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/
- https://en.wikipedia.org/wiki/State_diagram

Use case 04: Add Hostel


Filling Up the Hostel Adding Form
Hostel Adding Form Fill Up Completed
Requesting for adding the hostel
Requested For Add the Hostel
Approving by Admin
Hostel Adding Completed
Ready for post an ad
Requesting again for adding the Hosel
Rejecting by Admin
Recheck Done
Rechecking
Not Approved

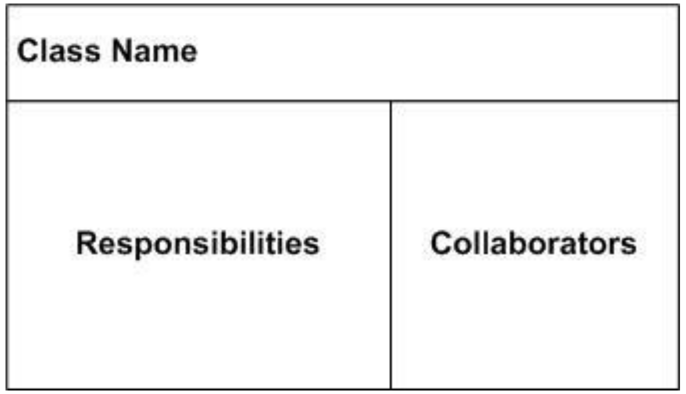
Use case 05: Ads. Post



CRC DIAGRAM

CRC Diagram – Class Responsibility Collaboration Diagram

It represents each classes responsibilities and with which class it collaborates to complete it's responsibilities.



Each CRC card has three components:

1. On top of the card, the **class** name
2. On the left, the **responsibilities** of the class
3. On the right, **collaborators** (other classes) with which this class interacts to fulfill its responsibilities

References:

- http://www.uml.org.cn/umlapplication/pdf/crcmodeling.pdf
- https://www.visual-paradigm.com/support/documents/vpuserguide/94/1289/6518_drawingcrcca.html
- https://en.wikipedia.org/wiki/Class-responsibility-collaboration_card

Advertisement	
• Providing Ads form • Ads posting • Showing Ads	- Hostel - User

Hostel	
• Providing Hostel Form • Hostel Adding • Show Hostel Details	- User

Complaint	
• Providing Complaint Form • Adding Complaint to DB • Show Complaint	- User

Review	
• Taking review from students • Adding reviews to DB • Show reviews	- Student - Hostel

E-R DIAGRAM

According to Wikipedia, an entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.
The entity-relationship (E-R) data model was developed to facilitate database design by allowing specification of an enterprise that schema represents the overall logical structure of a database.

Three basic concepts of E-R Diagram: **Entity sets**, **Relationship sets**, and **Attributes**

Entity Sets:
An entity is a **“thing”** or **“object”** in the real world that is distinguishable from all other objects.
For example, each person in a university is an entity. An entity is represented by a set of attributes. An entity set is a set of entities of the same type that share the same properties, or attributes. The set of all people who are Faculty Members of UIU, for example, can be defined as the entity set “Faculty_member”.

Relationship sets:
A relationship is an association among several entities. For example, we can define a relationship mentor that associates faculty member X and student Y. The relationship specifies that X is the mentor of student Y.

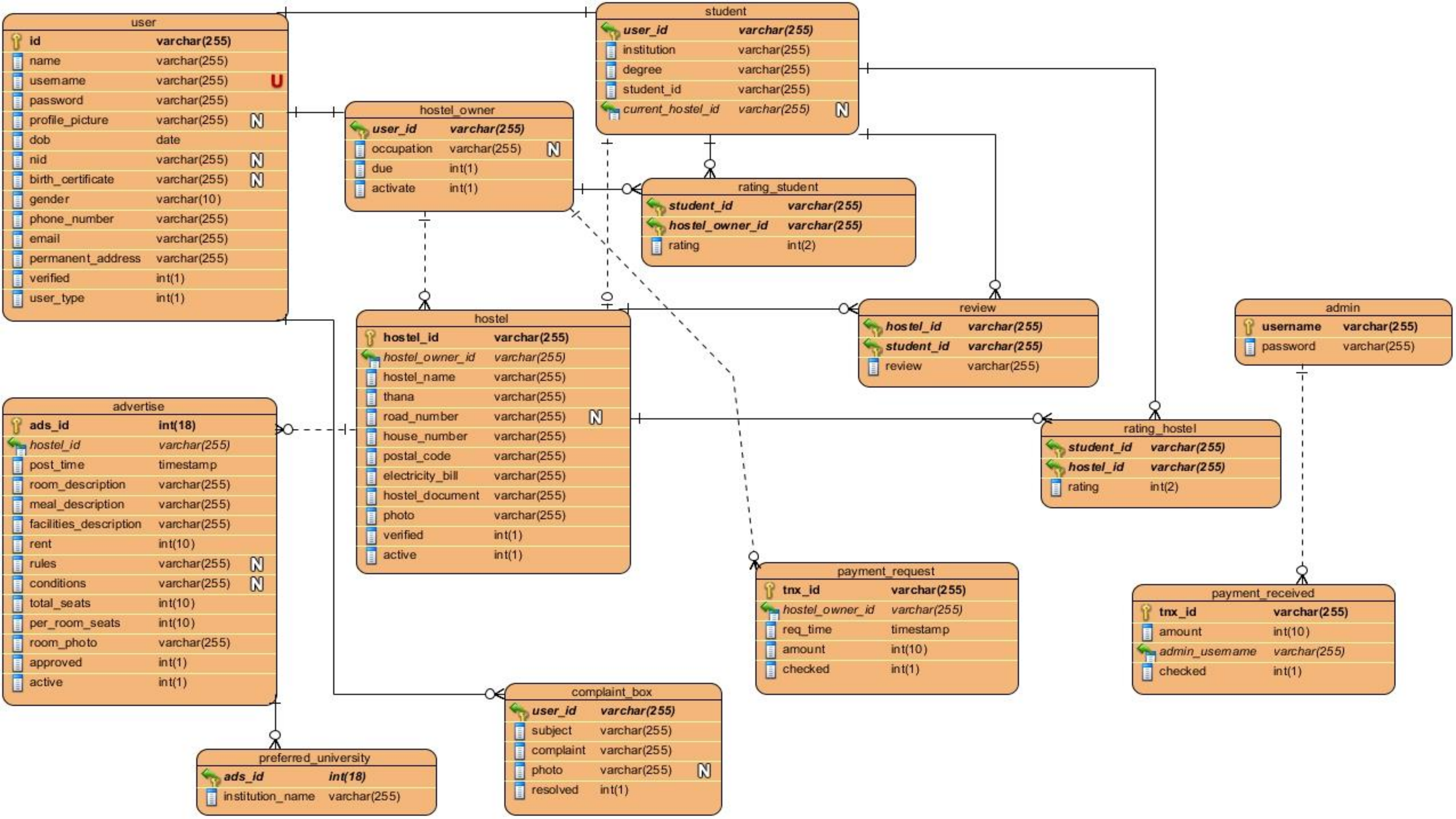
Attributes:
Entities are represented by means of their **properties**, called attributes. All attributes have values. For example, a student **entity** may have name, class, and age as attributes.

Entities we are using for our Project:

- ❑ User
- ❑ Hostel Owner
- ❑ Student
- ❑ Admin
- ❑ Hostel
- ❑ Preferred Institution
- ❑ Advertise
- ❑ Complaint Box
- ❑ Payment Request
- ❑ Payment Received
- ❑ Rating [Hostel]
- ❑ Review [Hostel]
- ❑ Rating [Student]

References:

- Database System Concepts 6th ed. Abraham Silberschatz, Henry F. Korth, S. Sudarshan
- https://www.tutorialspoint.com/dbms/er_diagram_representation.htm


user
id varchar(255)
name varchar(255)
username varchar(255) U
password varchar(255)
profile_picture varchar(255) N
dob date
nid varchar(255) N
birth_certificate varchar(255) N
gender varchar(10)
phone_number varchar(255)
email varchar(255)
permanent_address varchar(255)
verified int(1)
user_type int(1)
student
user_id varchar(255)
institution varchar(255)
degree varchar(255)
student_id varchar(255)
current_hostel_id varchar(255) N
hostel_owner
user_id varchar(255)
occupation varchar(255) N
due int(1)
activate int(1)
rating_student
student_id varchar(255)
hostel_owner_id varchar(255)
rating int(2)
hostel
hostel_id varchar(255)
hostel_owner_id varchar(255)
hostel_name varchar(255)
thana varchar(255)
road_number varchar(255) N
house_number varchar(255)
postal_code varchar(255)
electricity_bill varchar(255)
hostel_document varchar(255)
photo varchar(255)
verified int(1)
active int(1)
review
hostel_id varchar(255)
student_id varchar(255)
review varchar(255)
admin
username varchar(255)
password varchar(255)
advertise
ads_id int(18)
hostel_id varchar(255)
post_time timestamp
room_description varchar(255)
meal_description varchar(255)
facilities_description varchar(255)
rent int(10)
rules varchar(255) N
conditions varchar(255) N
total_seats int(10)
per_room_seats int(10)
room_photo varchar(255)
approved int(1)
active int(1)
rating_hostel
student_id varchar(255)
hostel_id varchar(255)
rating int(2)
payment_request
tnx_id varchar(255)
hostel_owner_id varchar(255)
req_time timestamp
amount int(10)
checked int(1)
payment_received
tnx_id varchar(255)
amount int(10)
admin_username varchar(255)
checked int(1)
complaint_box
user_id varchar(255)
subject varchar(255)
complaint varchar(255)
photo varchar(255) N
resolved int(1)
preferred_university
ads_id int(18)
institution_name varchar(255)

UI DESIGN

Definition

A user interface is the part of the system with which the users interact.

The user interface design defines the way in which the external entities will interact with the system and the nature of the inputs and outputs that the system accepts and produces.

Three Golden Rules of User Interface Design

1. **Place Users in Control**
   i. **Modeless**
   ii. **Flexibility**
   iii. **Interruptible**
   iv. **Helpful**
   v. **Forgiving**
   vi. **Navigable**
   vii. **Accessible**
   viii. **Facilitative**
   ix. **Preferences**
   x. **Interactive**

Golden Rules of User Interface Design

2. Reduce Users' Memory Load

- **i. Relieve short-term memory:**
- **ii. Rely on recognition, not recall**
- **iii. Provide visual cues:**
- **iv. Forgiving:**
- **v. Frequency:**
- **vi. Promote an object-action syntax:**
- **vii. Use real-world metaphors**
- **viii. User progressive disclosure**
- **ix. Organize**

Golden Rules of User Interface Design

3. Make the Interface Consistent

i. **Continuity:**
ii. **Maintain consistency within and across products**
iii. **Keep interaction results the same**
iv. **Provide aesthetic appeal and integrity**
v. **Encourage exploration**

Shneiderman’s Eight Golden Rule (1987)

1. **Strive for Consistency:**

   Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens;

   and consistent color, layout, capitalization, fonts, and so on, should be employed throughout.

2. **Seek universal usability:**

   Recognize the needs of diverse users and design for plasticity, facilitating transformation of content. Adding features for novices, such as explanations, and features for experts, such as shortcuts and faster pacing, enriches the interface design and improves perceived quality.

Shneiderman’s Eight Golden Rule (1987)

3. Offer informative feedback:

For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial.

4. Design dialogs to yield closure:

Informative feedback at the completion of a group of actions gives users the satisfaction of accomplishment, a sense of relief, a signal to drop contingency plans from their minds, and an indicator to prepare for the next group of actions.

5. Prevent errors:

Design the interface so that users cannot make serious errors, the interface should offer simple, constructive, and specific instructions for recovery.

Shneiderman’s Eight Golden Rule (1987)

6. Permit easy reversal of actions:

This feature relieves anxiety, since users know that errors can be undone, and encourages exploration of unfamiliar options.

7. Keep users in control:

Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. They don’t want surprises or changes in familiar behavior, and they are annoyed by tedious data-entry sequences, difficulty in obtaining necessary information, and inability to produce their desired result

Shneiderman’s Eight Golden Rule (1987)

8. Reduce short-term memory load:

Humans’ limited capacity for information processing in short-term memory (**the rule of thumb is that people can remember “seven plus or minus two chunks” of information**) requires that designers avoid interfaces in which users must remember information from one display and then use that information on another display.

It means that cellphones should not require reentry of phone numbers, website locations should remain visible, and lengthy forms should be compacted to fit a single display.

Norman’s Design Principles (1988)

1. Visibility: Can I see it?

Users need to know what all the options are, and know straight away how to access them. The clearer and visible functions are; the more likely users will be able to know what to do next.

2. Feedback: what is it doing?

Feedback is about sending back information about what action has been done and what has been accomplished, allowing the person to continue with the activity.

3. Affordance: Is it self descriptive?

Affordance is the link between how things look and how they’re used.

Norman’s Design Principles (1988)

4. Mapping: where am I and where can I go?

This refers to the relationship between controls and their effects in the world. Nearly all artifacts need some kind of mapping between controls and effects, whether it is a flashlight, car, power plant, or cockpit.

5. Constraints: why can’t I do that?

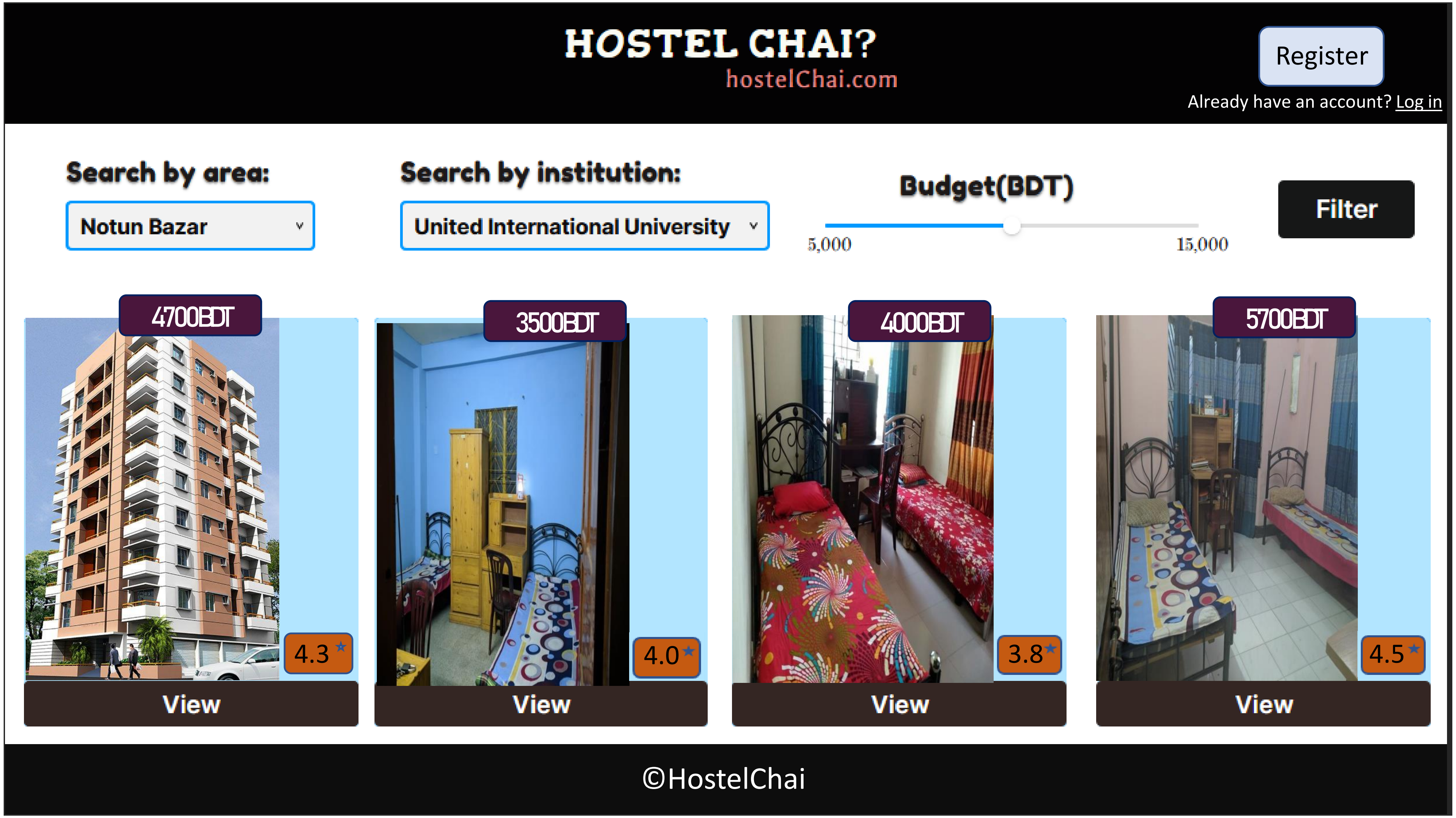
The design concept of constraining refers to determining ways of restricting the kind of user interaction that can take place at a given moment. There are various ways this can be achieved. A visual metaphor would be like a jail cell—people can still move around and interact, but they are confined to a certain area under certain parameters.

6. Consistency: Have I seen this before?

A consistent interface is one that follows rules, such as using the same operation to select all objects.

References

- **https://theomandel.com/resources/golden-rules-of-user-interface-design/**
- **https://theomandel.com/wp-content/uploads/2012/07/Mandel-GoldenRules.pdf**
- https://www.cs.umd.edu/users/ben/goldenrules.html
- https://www.inf.ed.ac.uk/teaching/courses/hci/1011/lecs/1_principles.pdf
- https://www.educative.io/edpresso/what-are-normans-design-principles
- https://laptrinhx.com/the-6-design-principles-of-don-norman-470095013/



HOSTEL CHAI?

hostelChai.com

Register

Already have an account? Log in

Search by area: Notun Bazar

Search by institution: United International University

Budget(BDT) 5,000 15,000

Filter

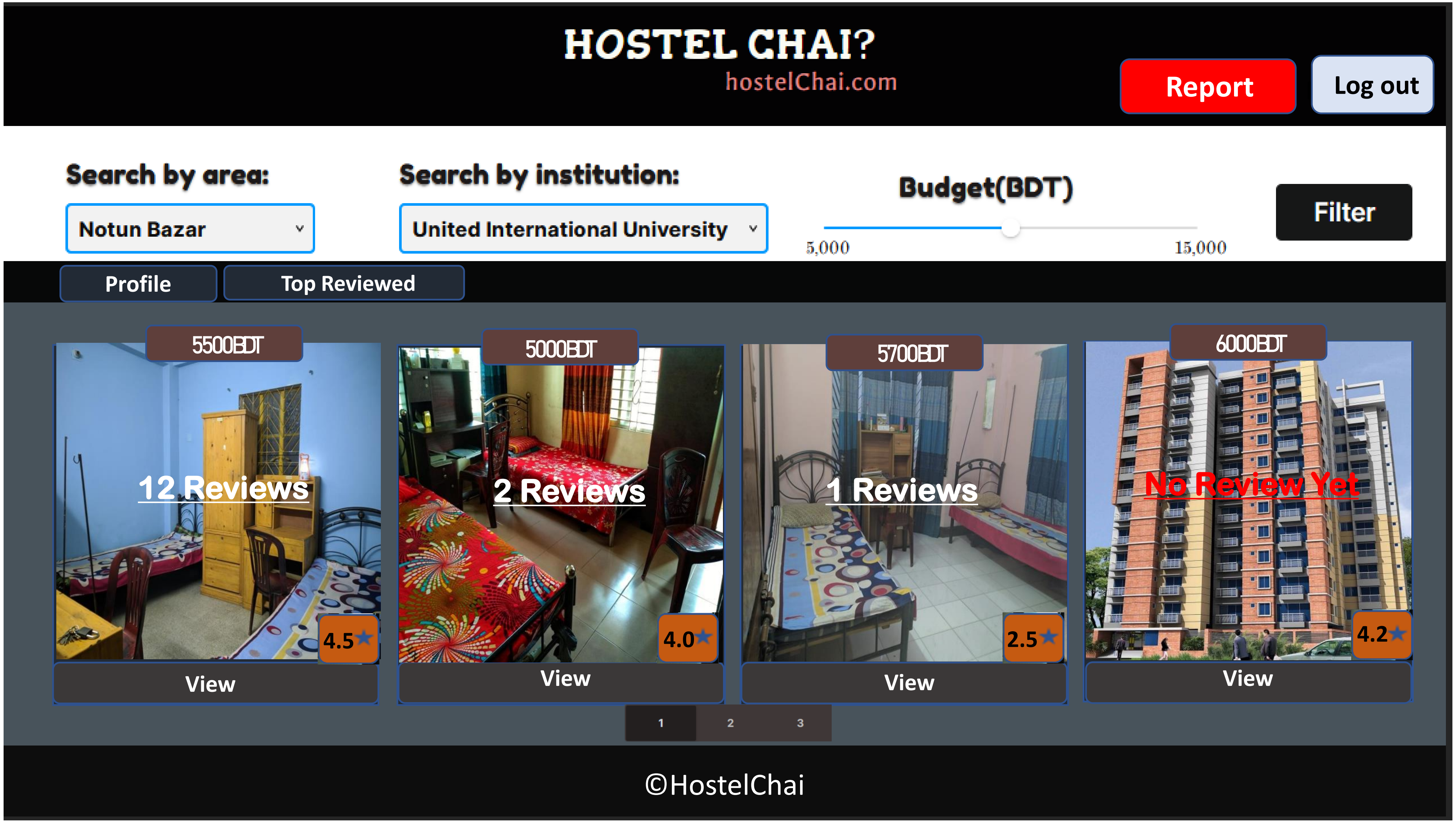
4700BDT 4.3★ View

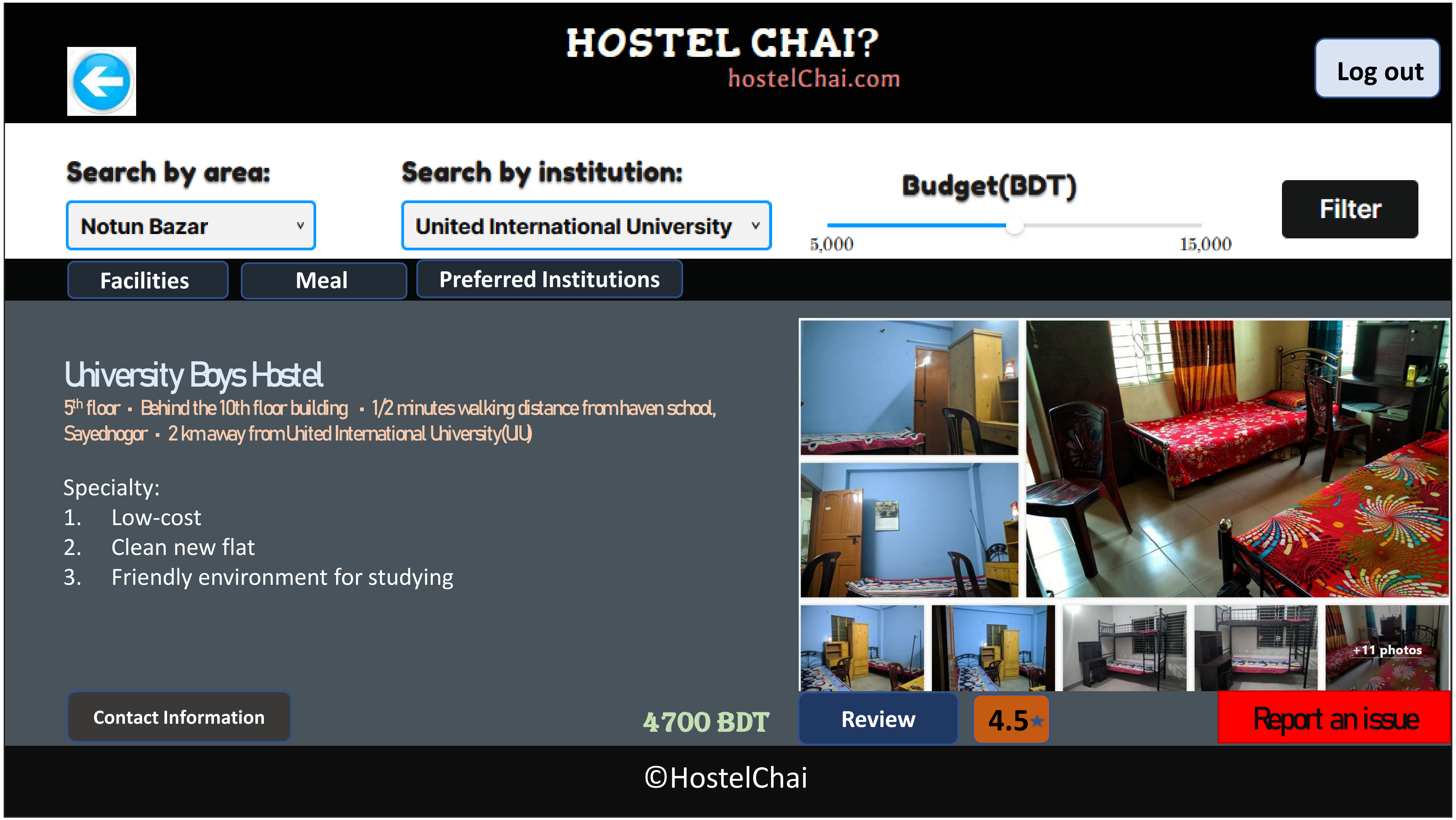
3500BDT 4.0★ View

4000BDT 3.8★ View

5700BDT 4.5★ View

©HostelChai


HOSTEL CHAI?
hostelChai.com
Report
Log out
Search by area:
Notun Bazar
Search by institution:
United International University
Budget(BDT)
5,000
15,000
Filter
Profile
Top Reviewed
5500BDT
12 Reviews
4.5
View
5000BDT
2 Reviews
4.0
View
5700BDT
1 Reviews
2.5
View
6000BDT
No Review Yet
4.2
View
1
2
3
©HostelChai



HOSTEL CHAI?

hostelChai.com

Log out

Search by area:

Notun Bazar

Search by institution:

United International University

Budget(BDT)

5,000

15,000

Filter

Facilities

Meal

Preferred Institutions

University Boys Hostel

5th floor · Behind the 10th floor building · 1/2 minutes walking distance from haven school, Sayednogor · 2 km away from United International University(UIU)

Specialty:

1. Low-cost
2. Clean new flat
3. Friendly environment for studying

+11 photos

Contact Information

4700 BDT

Review

4.5★

Report an issue

©HostelChai