Kingdom of Saudi Arabia
Ministry of Education
Qassim University
College of computer IT Department

Find Chalet

Prepared By:

Maha Alturaifi – 381213129
Leena Aldakheeli – 381210763
Kadi Alawaji-381211231
Renad Alsweed - 381210443

Table of Contents

INTRODUCTION

Most people tend to sit in a private and quiet place for entertainment, including chalets. In certain areas there are chalets for rent and people may want to rent them but don't know when they will be available for rent. Therefore, a mechanism is required to organize the process of renting a chalet, and this mechanism can be an application or website supported by the municipality or from a tourist area that includes a group of chalets for rent. In this report we will make an analysis of a system that enables the user to find a chalet that fits his needs and desires, along with what this includes of many Diagrams and requirements.

STAKEHOLDERS

- Chalet’s Owner
- Client
- System’s Admin

FUNCTIONAL REQUIREMENTS

- Client shall be able to log in to the system.
- Client shall be able to view all available chalets in the home page.
- Client shall be able to search about a specific chalet by its name.
- Client shall be able to search about chalets by time available (from time to time).
- Client shall be able to book a specific chalet.

- Client shall be able to cancel booking.
- Client shall be able to pay to confirm the process of booking.

- Chalet's Owner shall be able to log in to the system.
- Chalet's Owner shall be able to add one or more chalets information in the home page.
- Chalet's Owner shall be remove chalets information.
- Chalet's Owner shall be able to edit chalets information.

- Admin shall be able to log in to the system.
- Admin shall be able to edit chalets information.
- Admin shall be able to remove chalets information from the home page.

NON-FUNCTIONAL REQUIREMENTS

- **Look and feel requirements**
  - The system contains a unified user interface and easy to use.
- **Usability requirements**
  - The system contains direct functions that are easy to access by the user without conflicts.
- **Security requirements**

- The system keeps user data safe and not to exploit users' data or request permission from them that have nothing to do with the application's main functions.

- **Performance requirements**
    - The system should always function at the same level and provide responses to the user at a reasonable speed. This means no delay in returning information to the user and processing the data in the required time.
- **Portability requirements**
    - It is the possibility of transferring and developing the source code of the application to develop into other environments, whether at the level of hardware or software.

STRUCTURED SPECIFICATION

Function: Login
Description: The user of all three types, whether a client, owner of the chalet, or a system administrator wants to log in to the system to start his business, each according to his permissions.
Inputs:
1- User Name
2- Password

Outputs: User info and returning to homepage.
Pre-conditions: None
Action:
- The use-case starts when the user requests to login to the system.
- The system displays the login form to the user.
- The user enters his username and password.
- The system verifies the entered username and password.
- The system turns to home page of the system in the case of successful login and returns to the login page in the case of failed login.2

Post-conditions: The user of all three types, whether a client, a chalet owner, or a system administrator, goes to the home page of the system.
Side effects: None

Function: book a specific chalet
Description: The user of type client wants to book a specific chalet.
Inputs:
1- A specific chalet
2- Pay
Outputs: Completion of booking a chalet.
Pre-conditions: Login
Action:
- The use-case starts when the user requests to book a chalet.
- The system turns user to the chalet page which displays the chalet information including name, price, and available time.
- The user clicks the button “Confirm Booking”.
- The system turns user to payment page which displays the a payment gateway ex. PayPal.
- The system turns to home page of the system in the case of successful payment and returns to chalet page in the case of failed payment.

Post-conditions: if the process is successful the booking will be completed.
Side effects: None

Function: Add new chalet
Description: The user of type owner wants to add his new chalet to public to let clients later be able to book this chalet.
Inputs:
1- Chalet Name
2- Chalet Owner
3- Available times
4- Price
5- Location
Outputs: Completion of adding a new chalet.
Pre-conditions: Login with owner permission
Action:
- The use-case starts when the owner requests to add a new chalet.
- The system turns owner to the add chalet page which displays the form inputs including name, price, and all available time.
- The owner clicks the button “Add”.
- The system turns owner to his chalets list if any.

Post-conditions: Adding new chalet to DB and post it to home page.
Side effects: None

Function: delete chalet
Description: The user of type owner wants to delete a chalet from public page (home).
Inputs:
1- Chalet Id

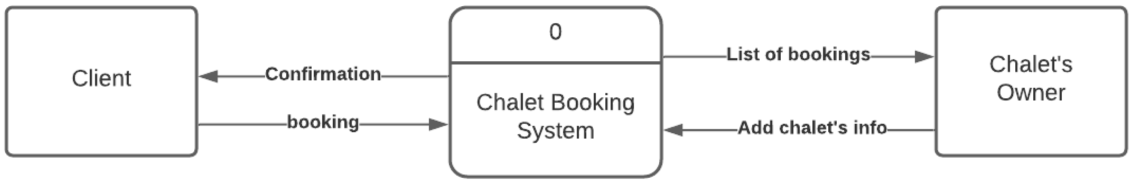
Outputs: Completion of deleting a chalet.
Pre-conditions: Login with admin or owner permission
Action:

- The use-case starts when the admin or owner requests to delete an existing chalet.
- The system asks admin or owner to the confirm deletion process completed.
- The owner clicks the button "yes".
- The system turns owner to his chalets list if any and the admin to home page.

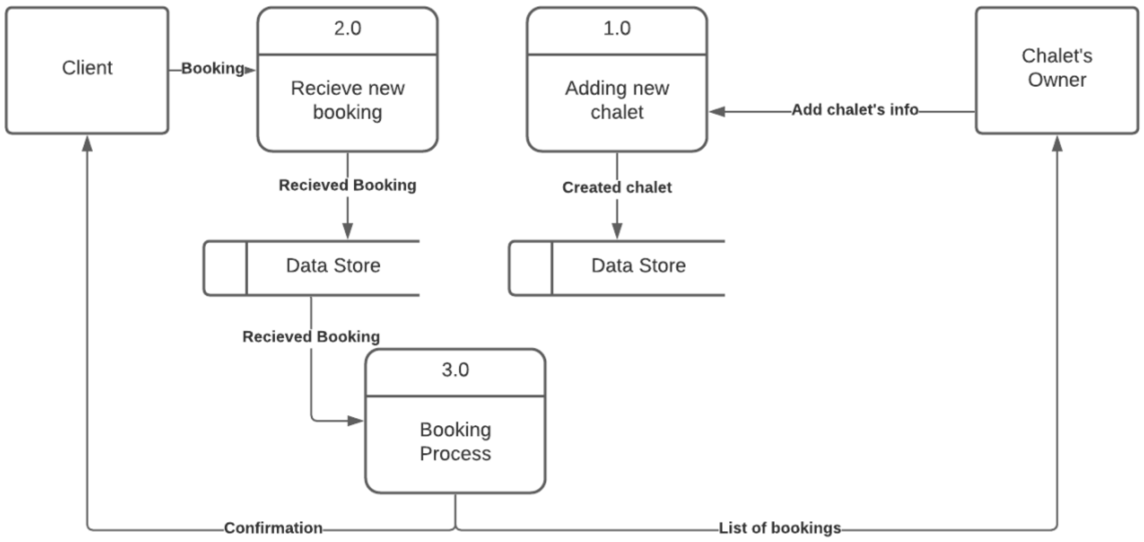
Post-conditions: Chalet will be deleted from both DB and home page.
Side effects: None

DATA FLOW DIAGRAM

1- DFD CONTEXT

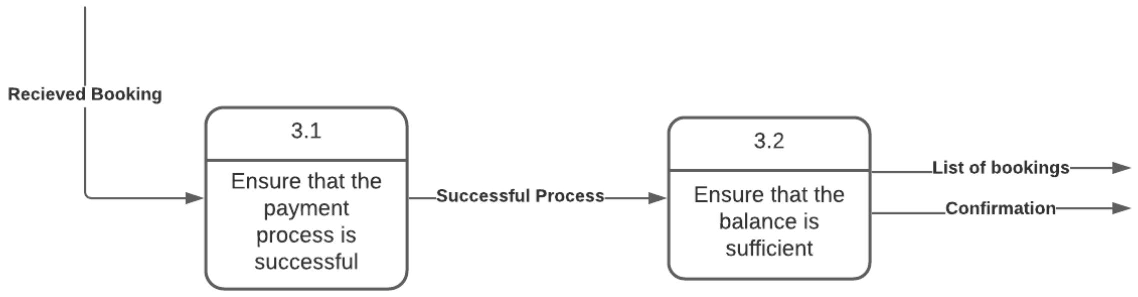


2- DFD LEVEL 0

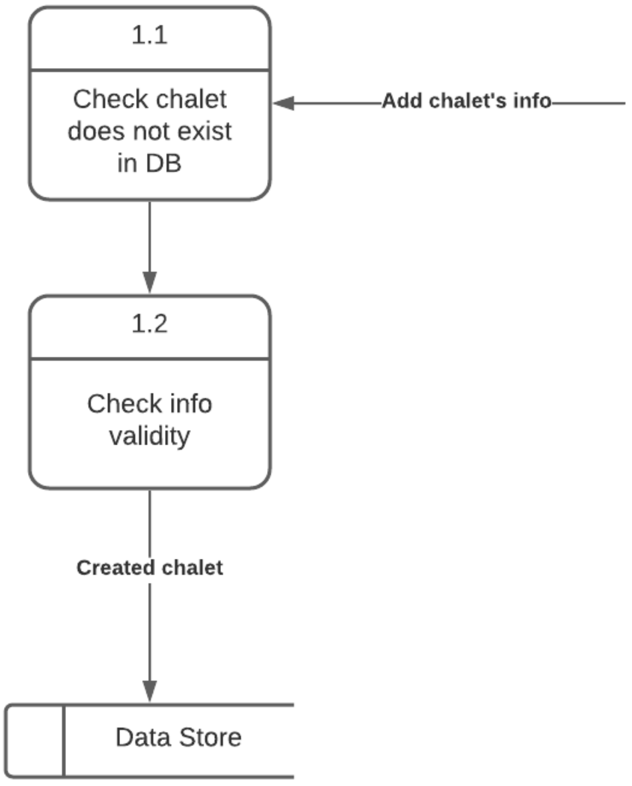


3- DFD LEVEL 1

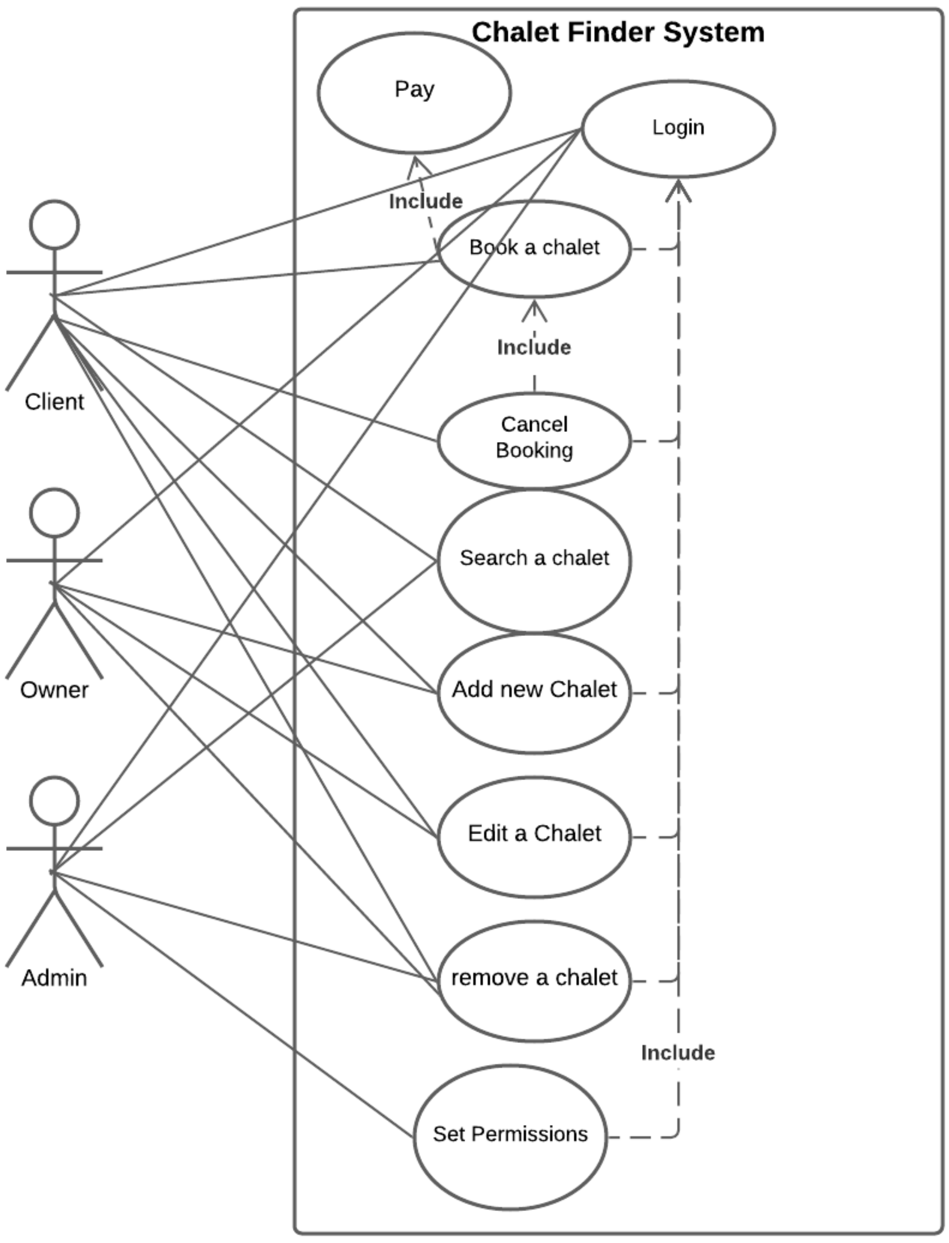
a. Process 3: Booking Process


Recieved Booking
3.1
Ensure that the payment process is successful
Successful Process
3.2
Ensure that the balance is sufficient
List of bookings
Confirmation

b. Process 1: Adding new chalet

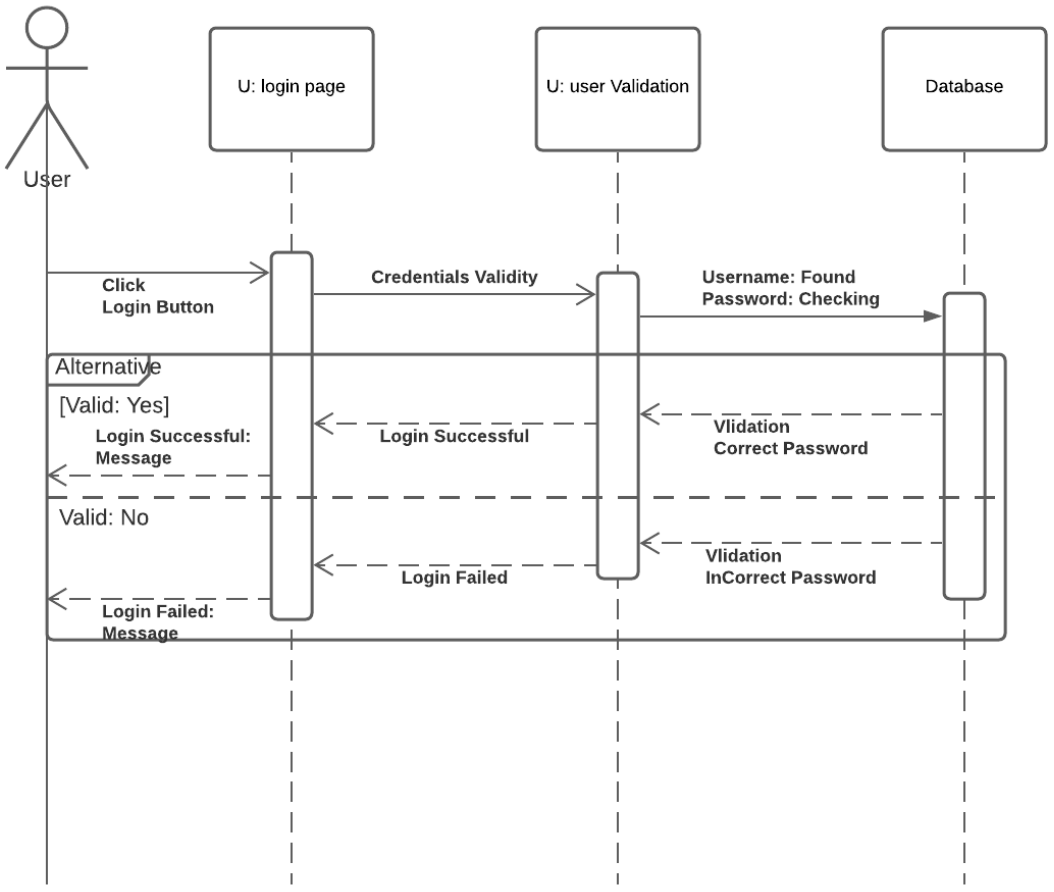


USE CASE DIAGRAM

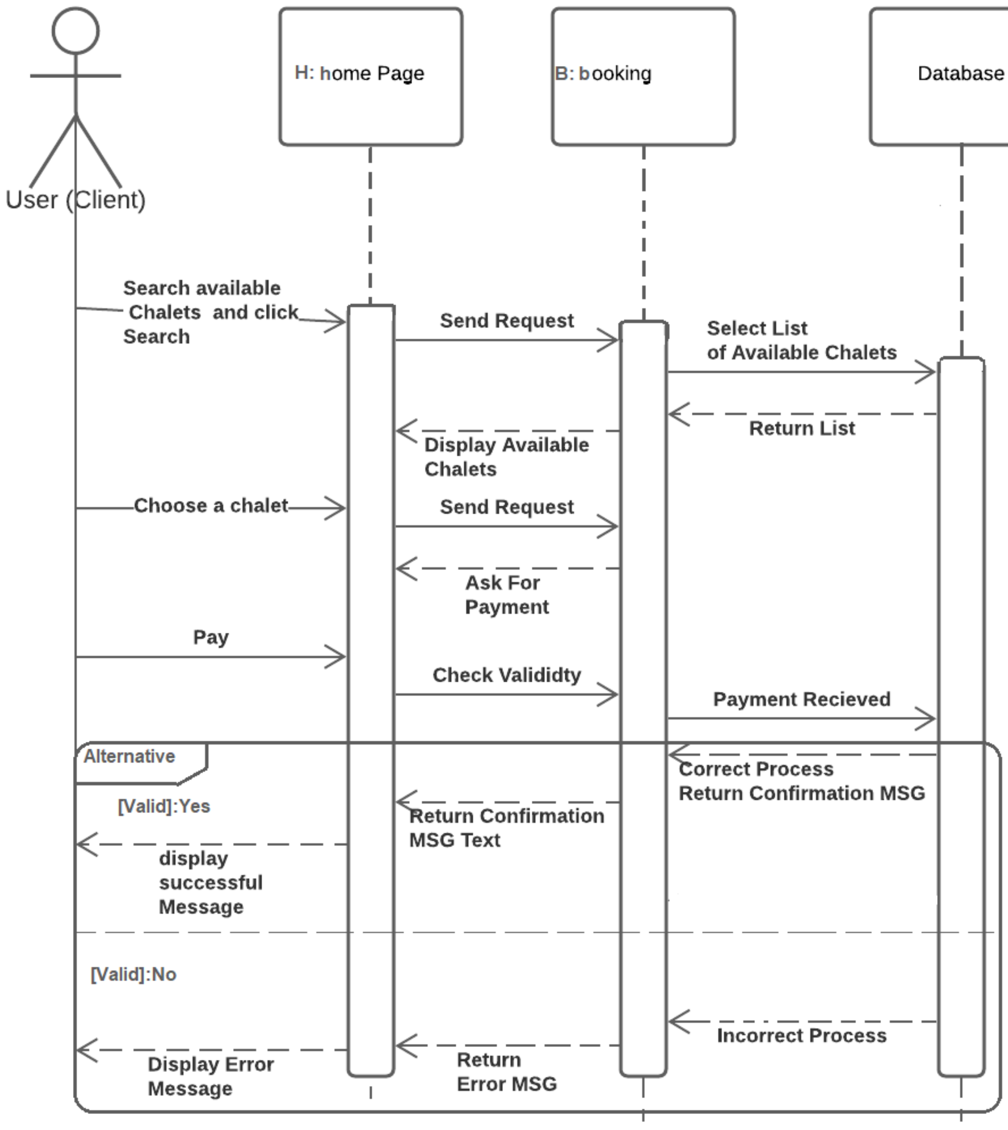


SEQUENCE DIAGRAM

Login Process

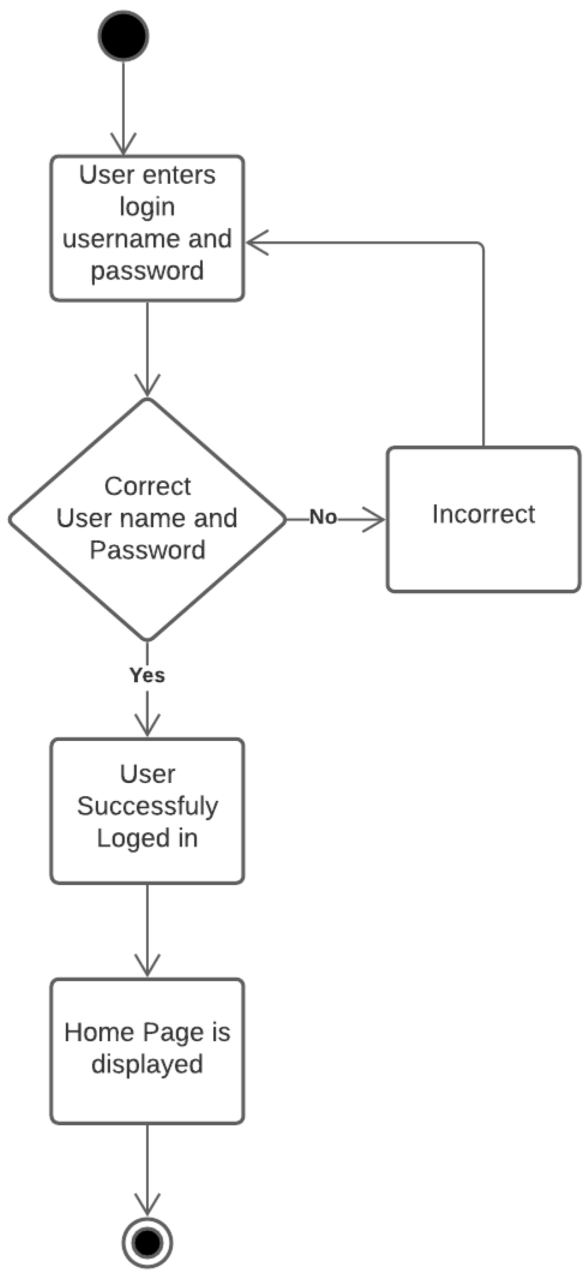


Booking Process



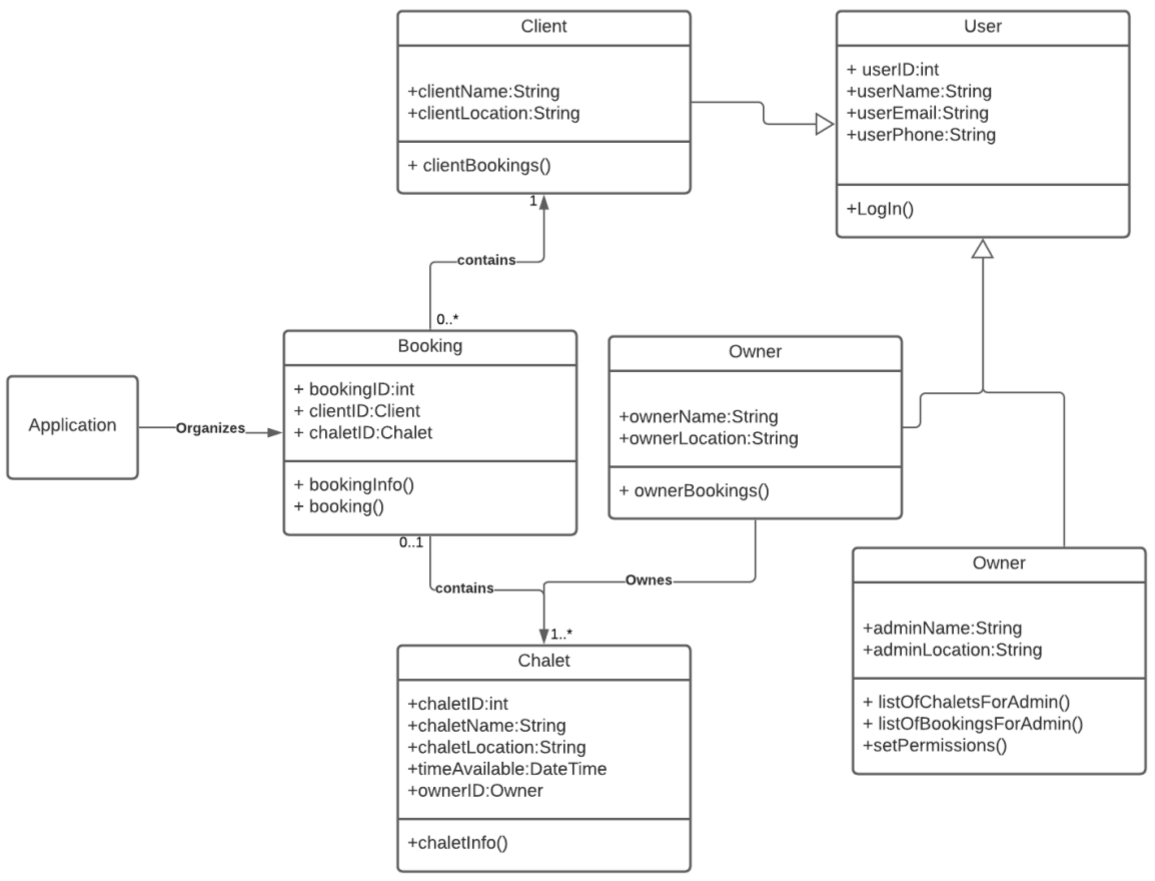
ACTIVITY DIAGRAM

Login Process



User enters login username and password
Correct User name and Password
No
Incorrect
Yes
User Successfuly Loged in
Home Page is displayed

CLASS DIAGRAM



ARCHITECTURAL PATTERN

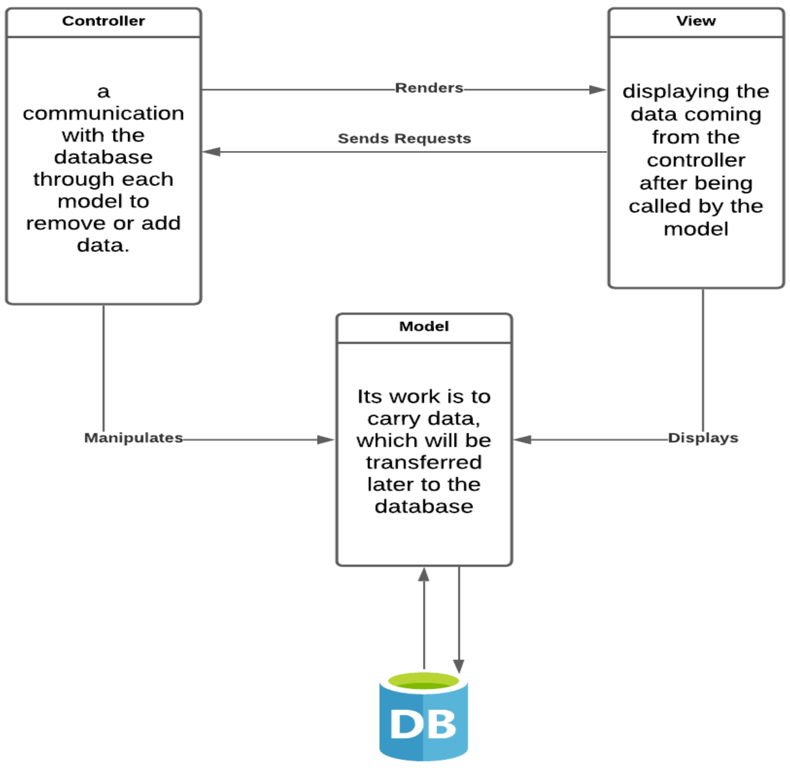
We will choose MVC, which is one of the most popular frameworks that organize software development. This is due to its ease of application and the unique way of dividing the code. The system is based on three pillars and the first letter of each pillar forms the MVC system, and they are Model, View, and Controller.

- The model is a normal class whose function is to carry data, which will be transferred later to the database.

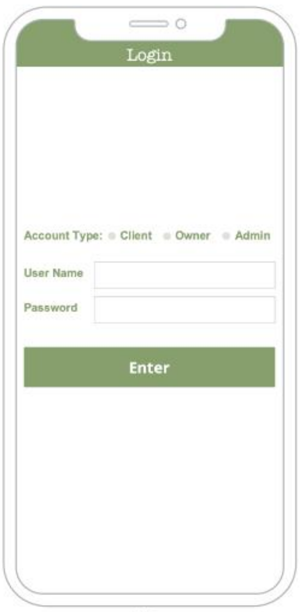
- The controller is also a class, but it is a little distinctive because it communicates with the database through each model to remove or add data.
- The view is a HTML page that will display the data coming from the controller after being called by the model (often it becomes a table in the Database).

And because the current project is a research project for chalets and after that a reservation is made for this chalet, we are dealing with a large amount of data, which must be programmatically managed in a distinctive way, so we suggested MVC to manage that effectively.

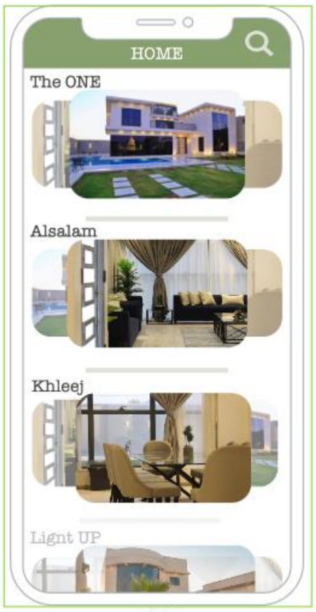
The following figure shows the MVC pattern.


Controller
a communication with the database through each model to remove or add data.
Renders
Sends Requests
View
displaying the data coming from the controller after being called by the model
Model
Its work is to carry data, which will be transferred later to the database
Manipulates
Displays
DB

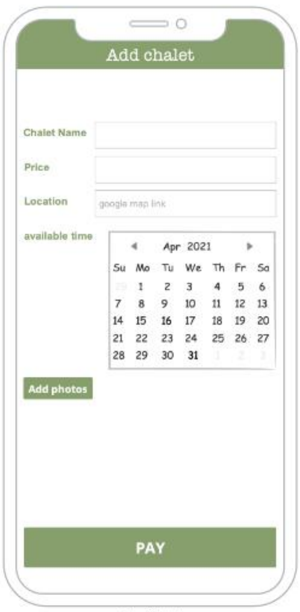
Interface



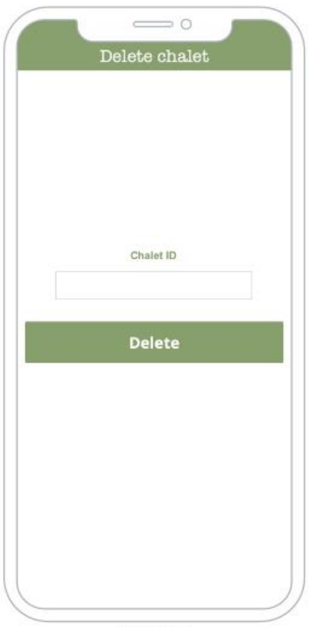
Login



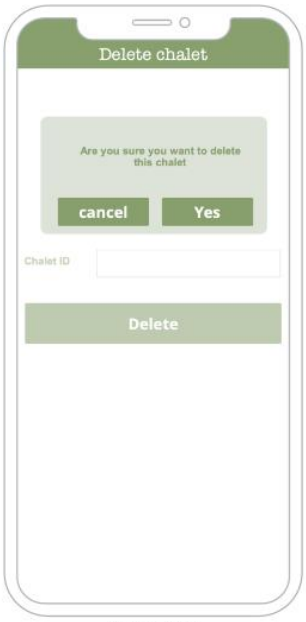
HOME



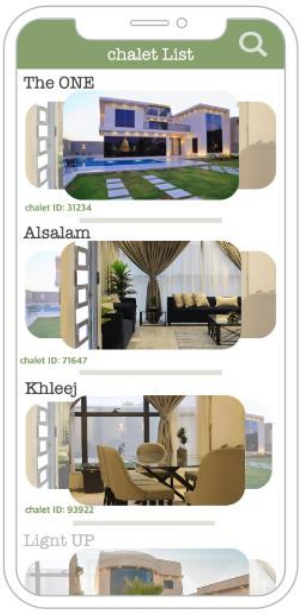
New Chalet



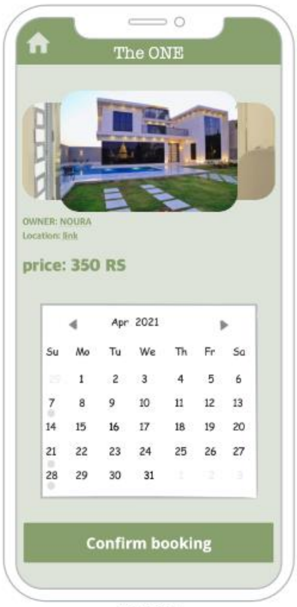
delete chalet



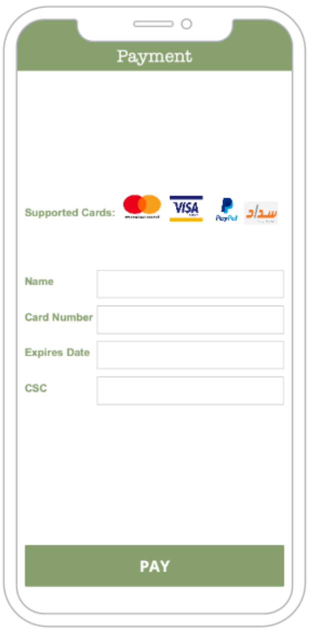
connfirm deletion



owner's chalet list



book chalet



payment