HBnB — Documentation Compilation

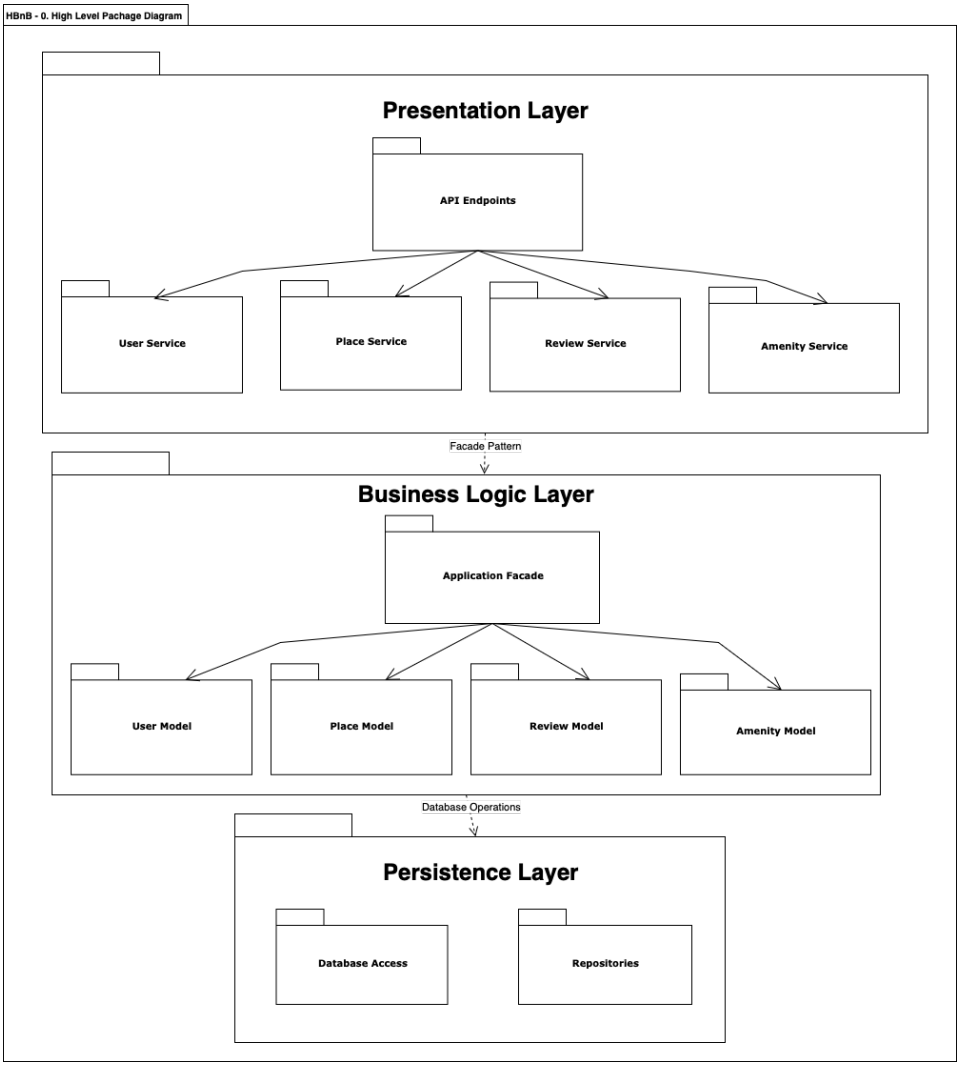
Introduction

This document presents the core architecture and API interaction flow of the HBnB Evolution application. HBnB Evolution application is a Holberton School project in which students design a simplified AirBnB-like application allowing users, both regular and administrators alike, to register and update their profiles. Users may also list the properties they wish to rent out, detailing the place in question and listing the amenities their accommodation has to offer. Renters may leave reviews of the properties which they have visited by rating them and commenting on their stay.

This document outlines how key features such as user registration, place creation, and review submission are handled across the Presentation, Business Logic, and Persistence layers of the application. The goal is to provide a clear structural overview to guide the implementation process. Diagrams are used to illustrate system design, responsibilities, and communication between each element.

High-Level Architecture

The HBnB Evolution application follows a layered architecture that separates processes across three distinct layers: Presentation, Business Logic, and Persistence. This multi-layered structure promotes scalability, and a clear distribution of responsibilities. We have chosen to use a High-Level Package Diagram to represent it. This diagram outlines the structural organisation of the system and the flow of data between each component across the three layers.



This package diagram encloses three packages detailing the inner workings of each of the three layers of HBnB Evolution application's architecture. It functions as follows:

Presentation Layer Package:

The Presentation Layer includes API Endpoints and service packages for users, places, reviews, and amenities. It handles HTTP requests, input validation, and structures a clear, readable output.

Business Logic Layer:

The Business Logic Layer contains the Application Facade which centralises and coordinates the API endpoints' access to the user, place, review and amenity models. The models themselves hold all the attributes and methods associated with their class.

Persistence Layer:

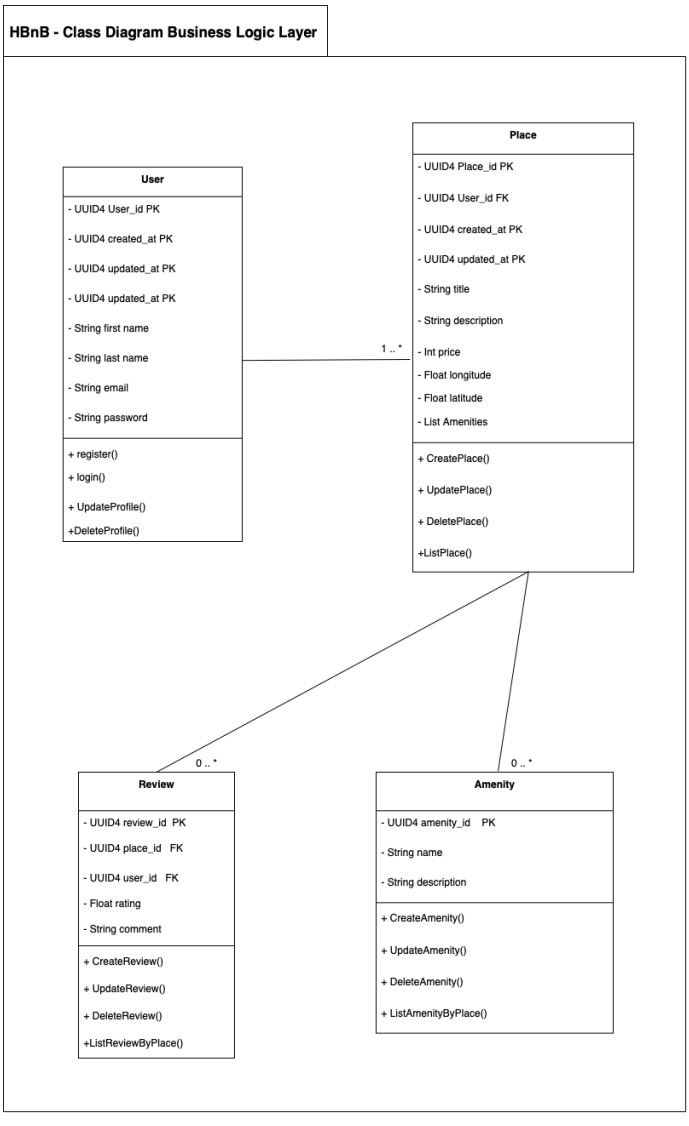
Both the Database Access and Repositories are enclosed in the Persistence Layer. This layer interacts directly with the database, thus allowing data storage and retrieval.

All layers communicate closely with their subsequent layer through different channels: the Facade Pattern and the Database Operations. The Facade pattern provides a unified and simplified interface between the Presentation and Business Logic layers by hiding the internal complexities and centralising access to core services. The Database Operations linking the Business Logic Layer and Persistence Layer carries a command from the models directly to the database.

This High-Level Package Diagram serves as the blueprint for system interactions, helping all API calls to flow more easily through the structure. It holistically illustrates the architecture structure and the different components that come into play in the HBnB Evolution application. The Presentation Layer handles incoming requests, which are then coordinated and enforced by the Business Logic Layer before being passed to the Persistence Layer for data storage or retrieval. Let us take a better look into the inner workings do the Business Logic Layer.

Business Logic Layer

The Business Logic Layer manages the rules and decision-making processes of the HBnB application Evolution as well as the manner in which all entities interact with one another. This Detailed Class Diagram illustrates the internal structure of this layer, mapping out the relationships between entities and the logic of how data is processed before reaching the database.



This class diagram shows how the Business Logic Layer is categorised by four different model classes. It functions as follows:

User Class:
This class contains all of the user related attributes and methods. It is key to creating and updating an account as well as accessing all user information. It also contains methods allowing user to register, login, as well as updating and deleting their profiles.

Place Class:
The place class contains all of the place related attributes and methods. It holds information like the location, price, amenities and the description of the place. Its methods let the user create, update, delete and list a place on the application.

Review Class:
The review class contains the rating and comment fields. The methods in this class allow the user to create, update, delete and list their review.

Amenity Class:
The place class contains all optional features linked to the rental places. It includes the amenity name and description with methods to create, update, delete, and list them.

Both the Review and Amenity Classes are associated with the place class of course. There can be no review or amenity can be listed without being tied to one or more places. The Place Class is then in turn associated to the User Class as a place must be owned by a user of the application. A user renting out a place may rent out a single place or many if they so wish.

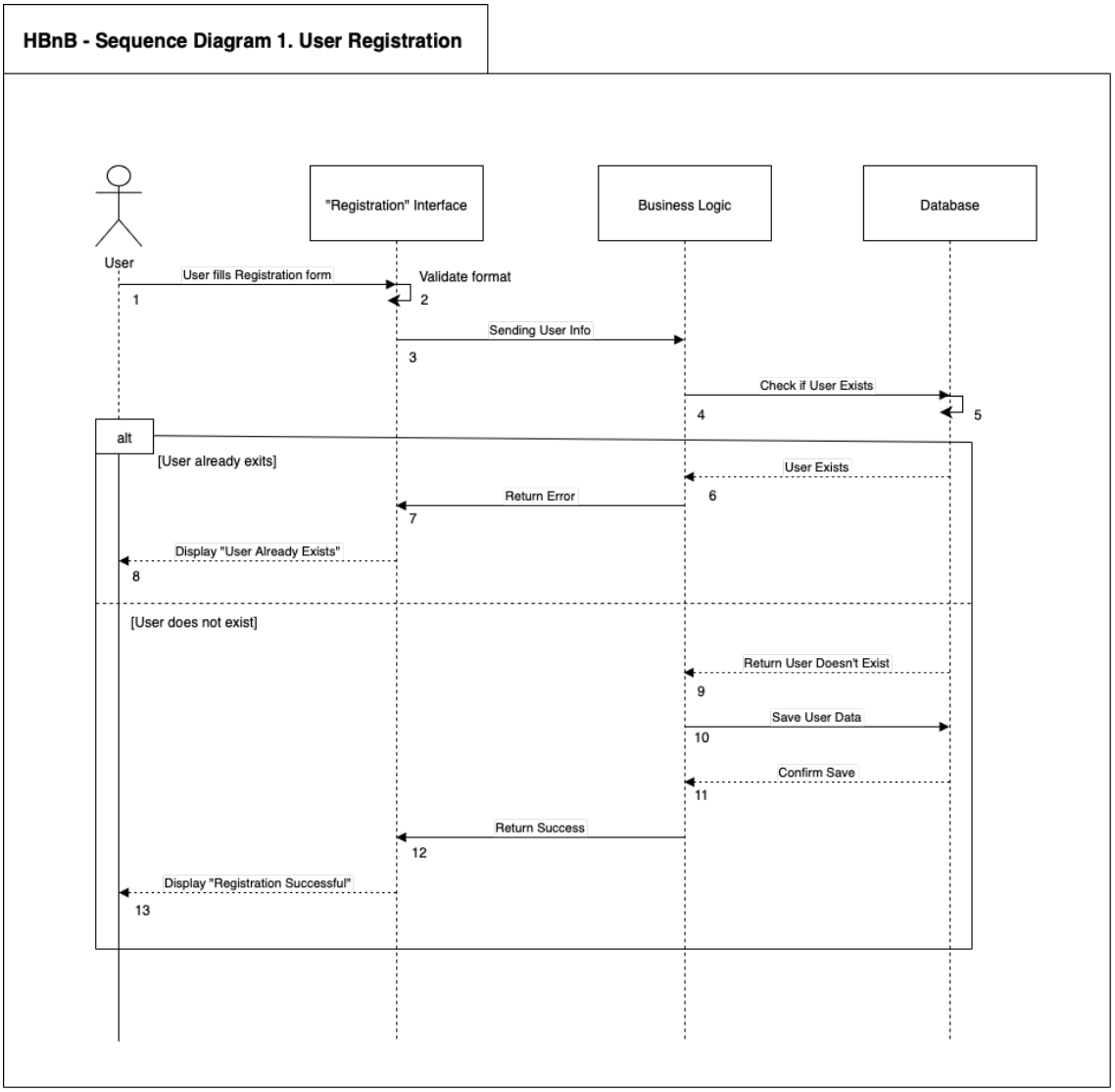
This diagram illustrates the relationship between all entities of the Business Logic Layer. They all constitute a whole that processes input, enforces rules, and prepares outputs before interacting with the Persistence Layer.

API Interaction Flow

In this section we will explore the API integration flow for the four categories mentioned above: user, place, review and amenity. We intend to illustrate the user registration, new place creation, review submission and the place search by filter using sequence diagrams to properly demonstrate fully the proceedings of these processes.

1. User registration

In order to properly use the HBnB Evolution application and be able to either book a reservation or rent out a property, one must first register as a user.



This sequence diagram demonstrates the process of registering as a new user to the HBnB Evolution application. It functions as follows:

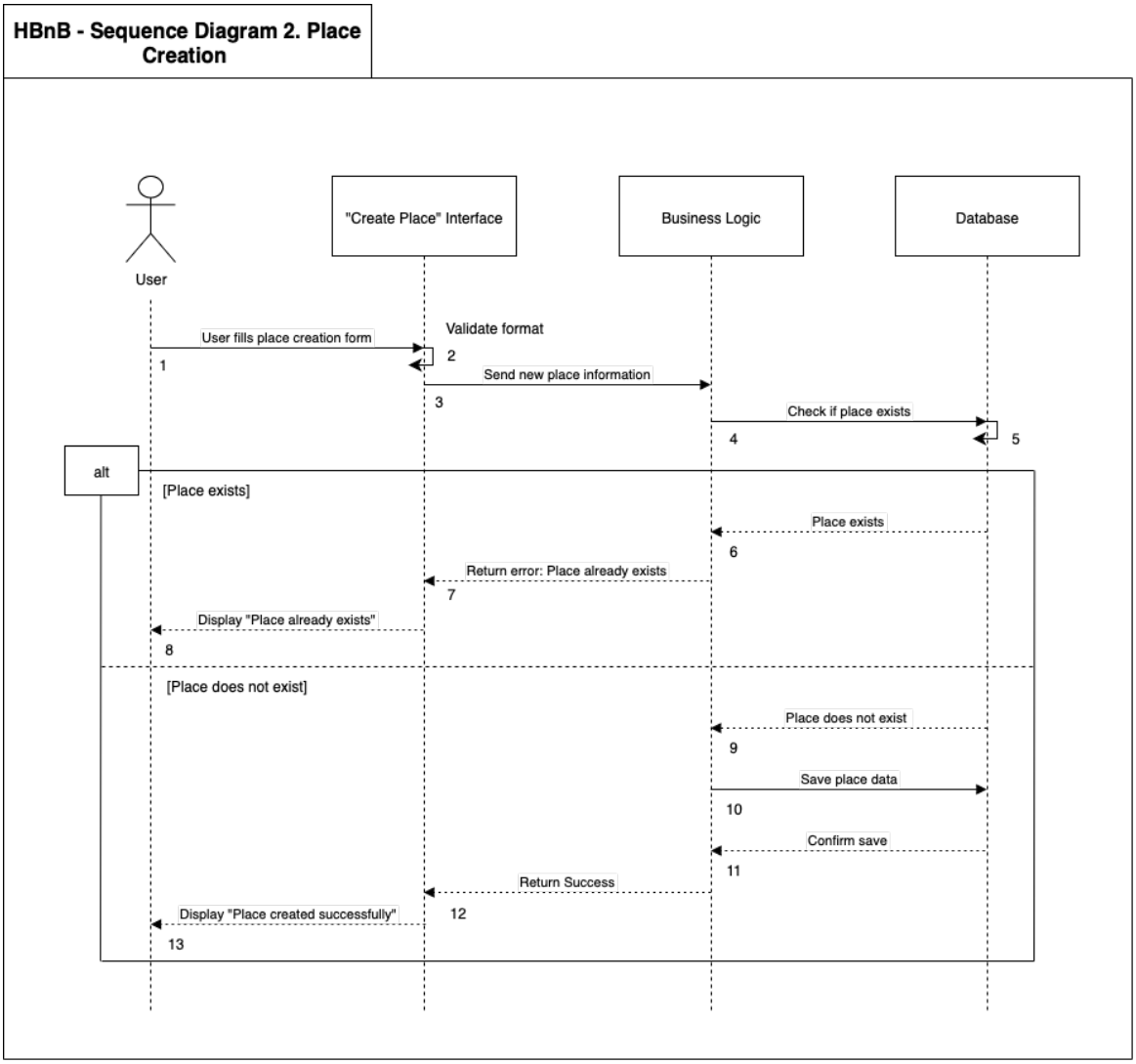
In this diagram, we can see one actor, the user, and three different objects: the “Registration” Interface, the Business Logic and the Database. The "Registration" Interface receives and validates user input before forwarding it to the Business Logic, which handles decision-making before interacting with the Database, which stores or retrieves the relevant user data to finalise the registration process.

The user starts by filling out their personal details on the registration form appearing on their browser, before that information is checked by the “Registration” Interface. Once the format has been validated, the user’s information is passed on to the Business Logic Layer where, as seen in the Business Logic Layer section, it passes by the user model and is sent off to the Database to verify whether this data already exists. If the user has previously already created a profile with this information, the Database will send the message back to the Business Logic layer which will relay

that message to the “Registration” Interface. The “Registration” Interface will then send out a message to the user interface displaying “User Already Exists” for the user to see. If, however, the Database shows that the data provided by the user is unique, it will be saved and stored in the Database. The Database will then send out a message confirming it has saved the user’s information to the Business Logic which will in turn transmit it to the “Registration” Interface. The “Registration” Interface will then send out a message displaying “Registration Successful” to the user.

2. Place Creation

Once a new user has registered successfully to the HBnB Evolution application, they might want to add their property to the listings in which case they must create a new place.



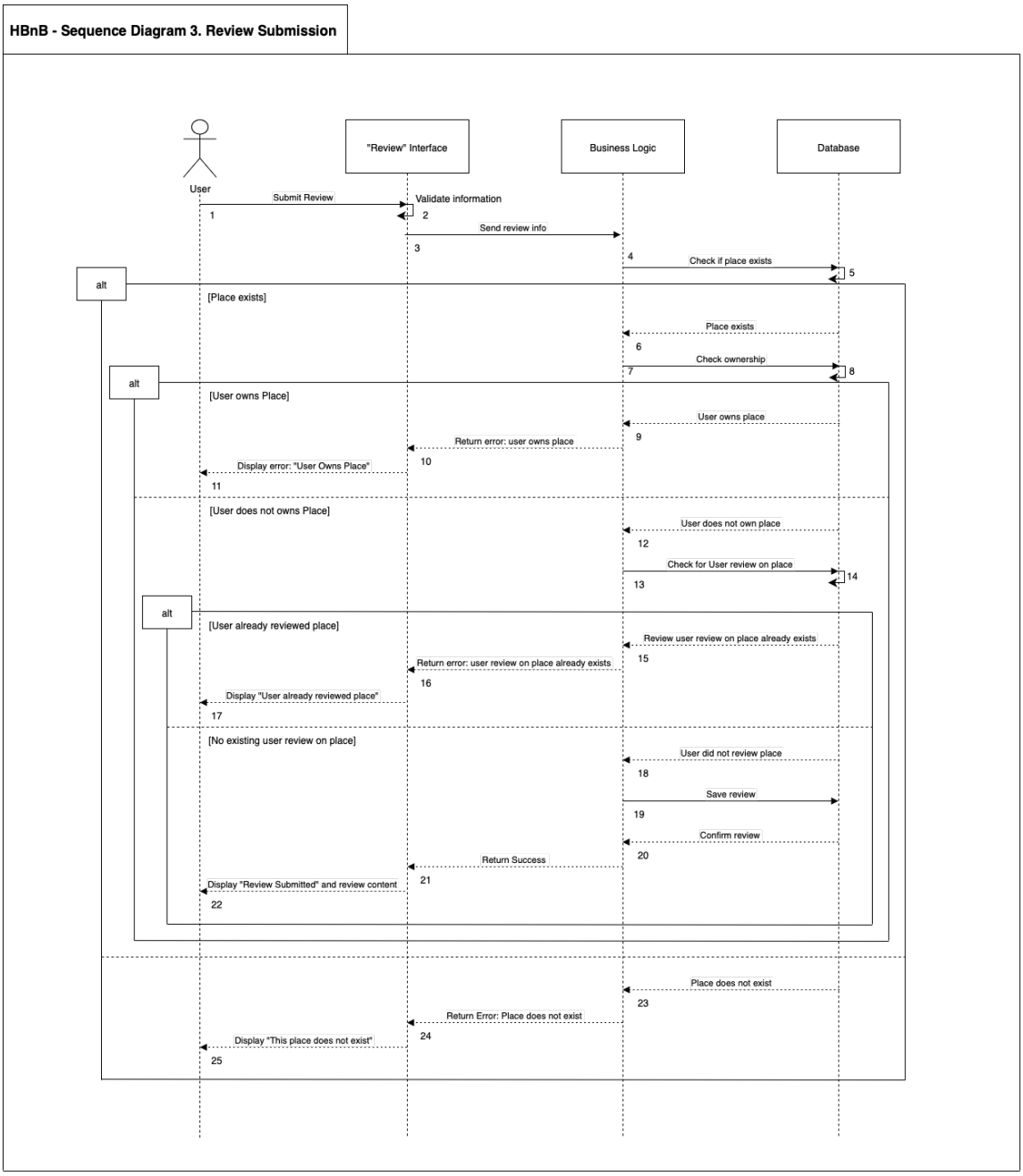
This sequence diagram demonstrates the process of creating a new place in the HBnB Evolution application. It functions as follows:

In this diagram, we can see one actor, the user, and three different objects: the “Create Place” Interface, the Business Logic and the Database. These objects all function and interact in the same way as they did in the previous diagram. The “Create Place” Interface receives and validates user input before forwarding it to the Business Logic, which handles decision-making before interacting with the Database, which stores or retrieves the relevant user data to finalise the registration process.

The user starts by filling out information related to their property on the form that appears on their browser, before that information is checked by the “Create Place” Interface. Once the format has been validated, the place’s information is passed on to the Business Logic Layer where, as seen in the Business Logic Layer section, it passes by the place model and is sent off to the Database to verify whether this data already exists. If the user has already created a listing for this property previously, the Database will send the message back to the Business Logic layer which will relay that message to the “Create Place” Interface. The “Create Place” Interface will then send out a message to the user interface displaying “Place Already Exists” for the user to see. If, however, the Database shows that the data provided by the user is unique, it will be saved and stored in the Database. The Database will then send out a message confirming it has saved the new place’s creation to the Business Logic which will in turn transmit it to the “Create Place” Interface. The “Create Place” Interface will then send out a message displaying “Place created successfully” to the user.

3. Review Submission

After their stay at an HBnB place, user might want to review the property by rating it and leaving a comment.



This sequence diagram demonstrates the process of leaving a review of a place in the HBnB Evolution application. It functions as follows:

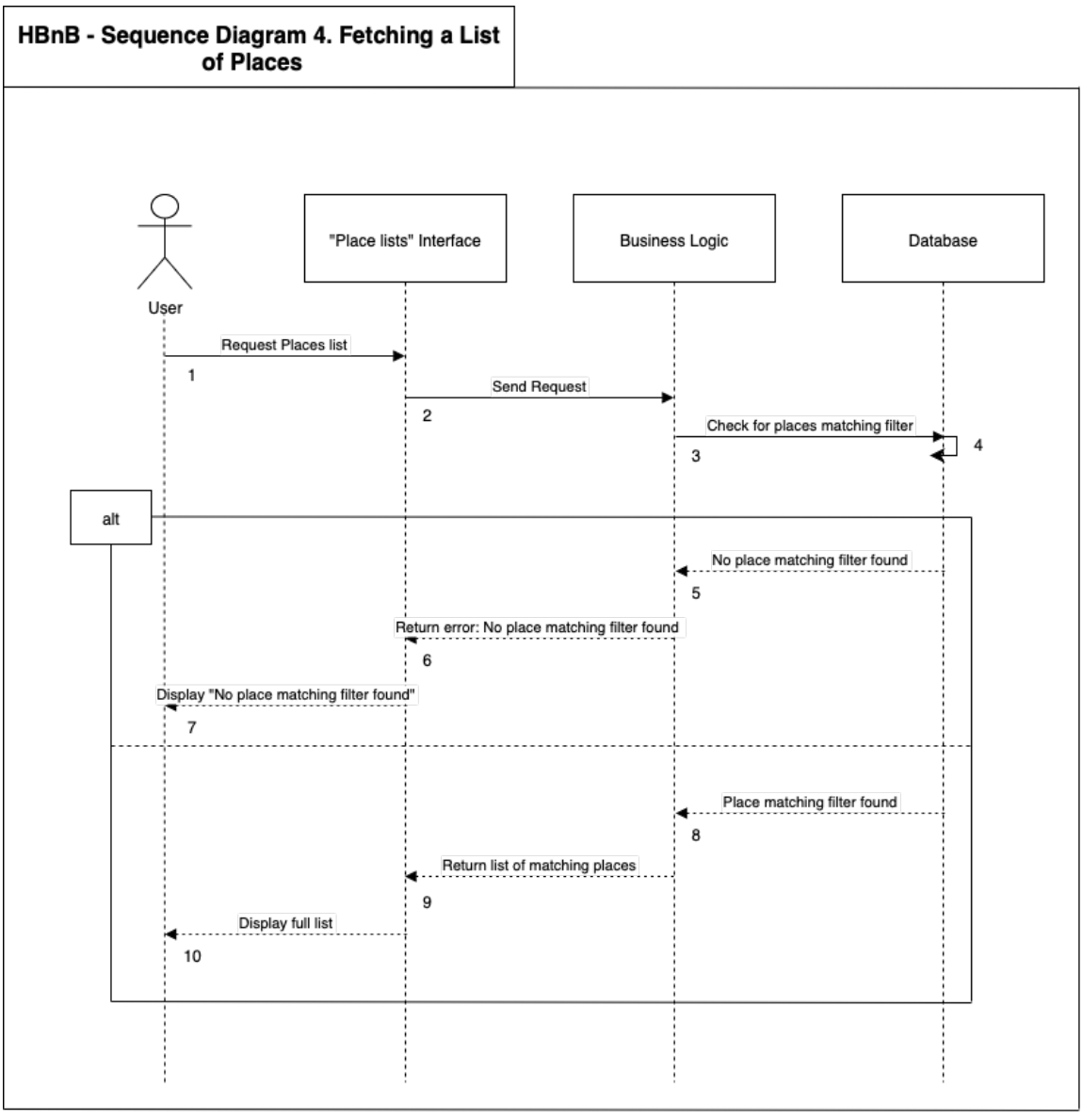
In this diagram, we can see one actor, the user, and three different objects: the “Review” Interface, the Business Logic and the Database. These objects all function and interact in the same way as they did in the previous diagrams.

The user starts by filling out the rating and comment of their review on the form that appears on their browser, before that information is checked by the “Review” Interface. Once the format has been validated, the review’s information is passed on to the Business Logic Layer where, as seen in the Business Logic Layer section, it passes by the review model and is sent off to the Database to verify whether the place being reviewed exists. If the place exists, the Business Logic Layer will send out a request to the Database to verify that the user does not own the place. If it turns out that

they do, a message will be returned and transmitted until the “Review” Interface sends out the message “User Owns Place.” If the user does not own the place, the Business Logic will again send out that information to the Database to check whether the user has already reviewed it. If so, the same messaging structure will relay the information back to the user with “User already reviewed place”. If the review is valid, the review’s rating and comment are saved and stored in the Database. The Database will then send out a message confirming it has saved the review to the Business Logic which will in turn transmit it to the “Review” Interface. The “Review” Interface will then send out a message displaying “Review Submitted” as well as its rating and contents to the user. If, however, the place being reviewed does not exist, the Database will send the message back to the Business Logic layer which will relay that message to the “Review” Interface. The “Review” Interface will then send out a message to the user interface displaying “This place does not exist” for the user to see.

4. Fetching a List of Places

In order to find a place suited to their liking, a user might want to filter their search while looking for a property to book.



This sequence diagram demonstrates the process of doing a filtered search on HBnB Evolution application browser interface. It functions as follows:

In this diagram, we can see one actor, the user, and three different objects: the “Place lists” Interface, the Business Logic and the Database. These objects all function and interact in the same way as they did in the previous diagram. The “Place lists” Interface receives and validates user input before forwarding it to the Business Logic, which handles decision-making before interacting with the Database, which stores or retrieves the relevant user data to finalise the registration process.

The user starts by filling out information related to the kind of property they are looking for and apply filters to the form that appears on their browser, before that information is checked by the “Place lists” Interface. The place’s information is then passed on to the Business Logic Layer where, as seen in the Business Logic Layer section, it passes by the amenities and/or place model depending on the user’s criteria, and is sent off to the Database to verify whether this data exists. If there exists no place matching their filters, the Database will send the message back to the Business Logic layer which will relay that message to the “Place lists” Interface. The “Place lists” Interface will then send out a message to the user interface displaying “No place matching filter found” for the user to see. If, however, the Database shows that the kind of place the user is looking for can be found, the data will be retrieved by the Database which will then send out the full list of places to the Business Logic. The Business Logic will in turn transmit the list to the “Place lists” Interface which will then send it to the user interface.