**Group Report**

Advanced Web Project

**Subject Title:** Advanced Web Project

**Team Name:** *HHTDB*

**Submission Date:** *08/07/2023*

**Team Information**

|  |  |
| --- | --- |
| FULL NAME | EMAIL |
| Phan Thị Thu Hương | Huong.phan24@student.passerellesnumeriques.org |
| Trần Quốc Hữu | Huu.Tran24@student.passerellesnumeriques.org |
| Phan Trần Thị Anh Thư | Thu.Phan24@student.passerellesnumeriques.org |
| Hồ Thị Bích | Bich.Ho24@student.passerellesnumeriques.org |
| Hồ Văn Đi | Di.Ho24@student.passerellesnumeriques.org |

2. Design Document

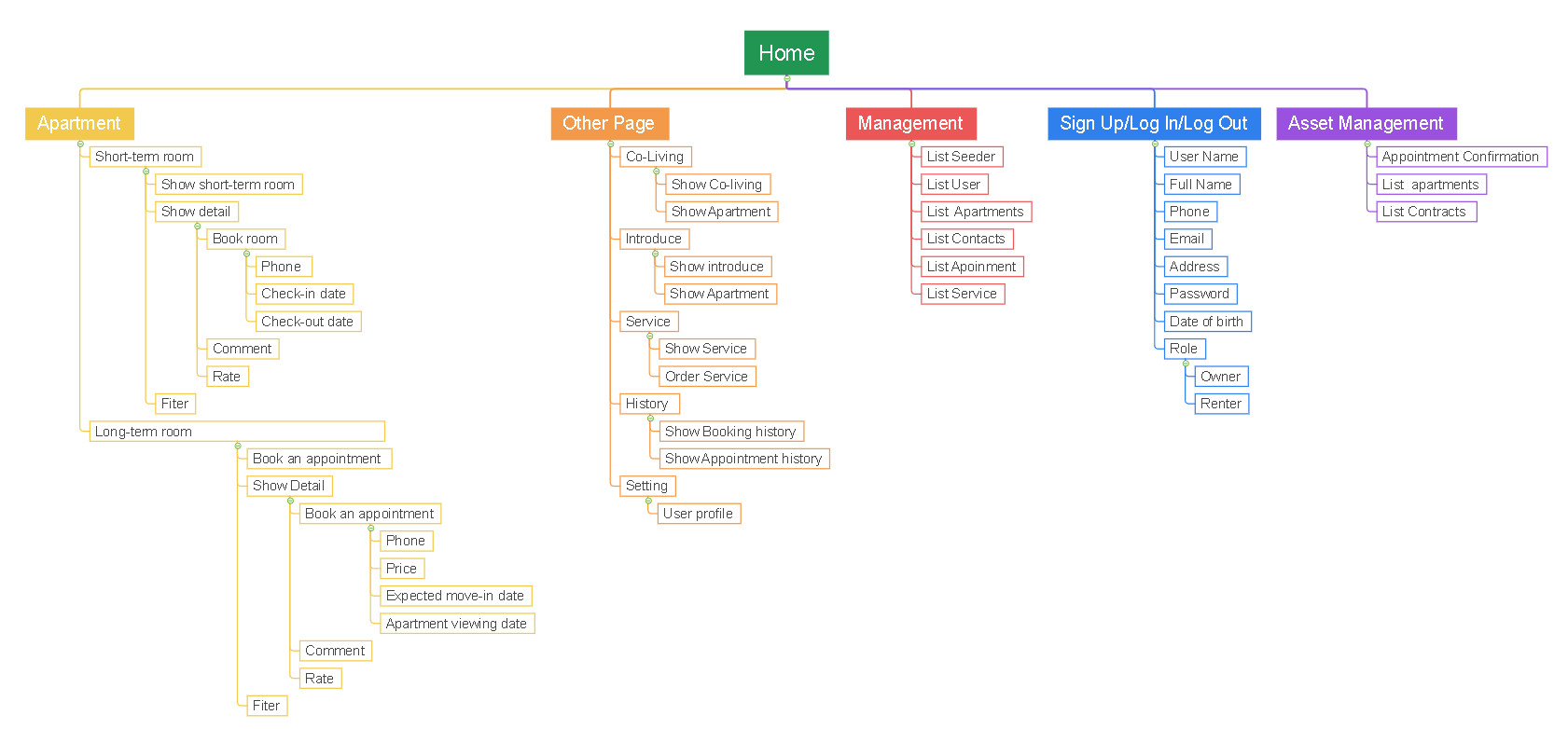
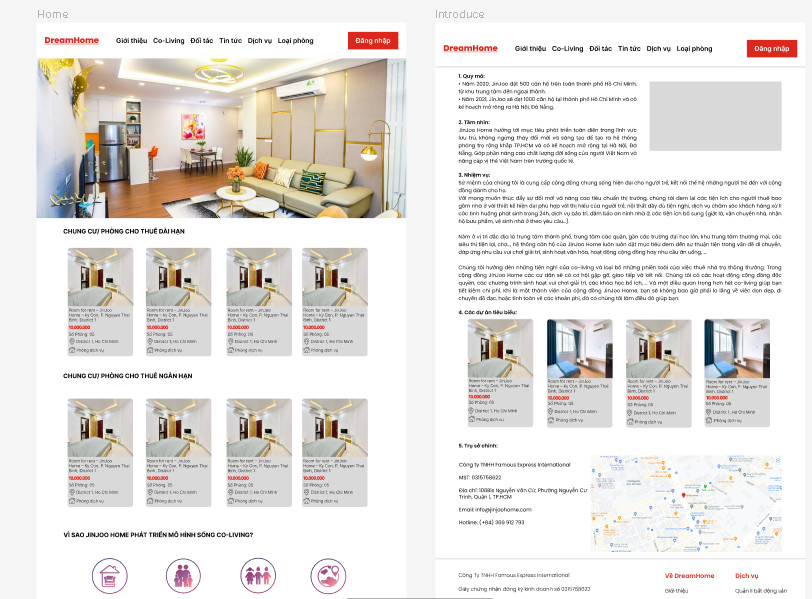
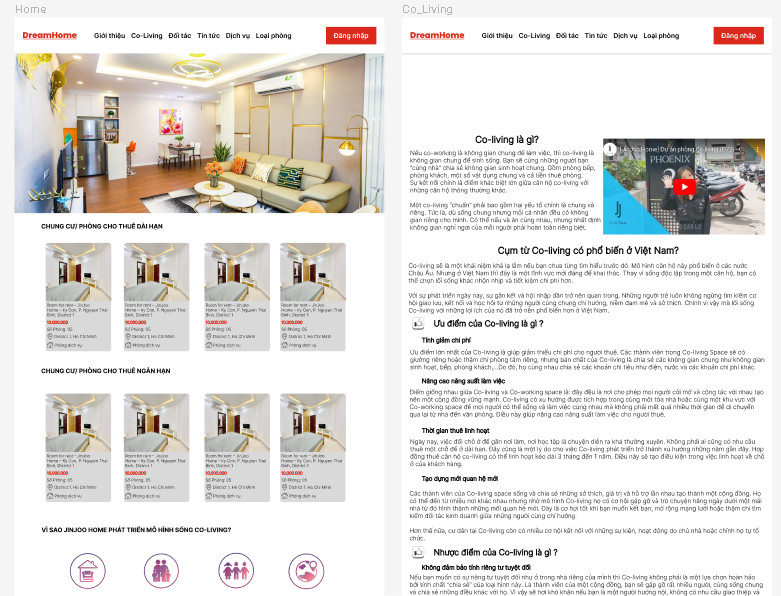
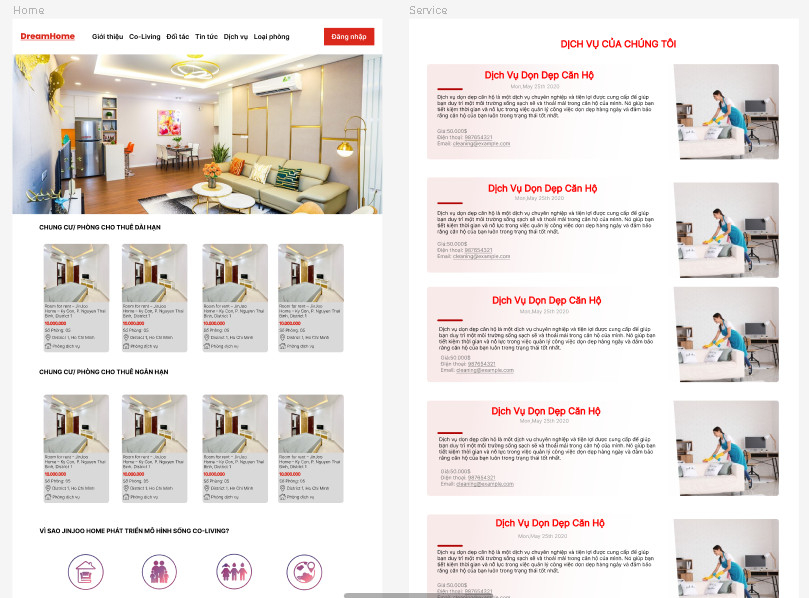
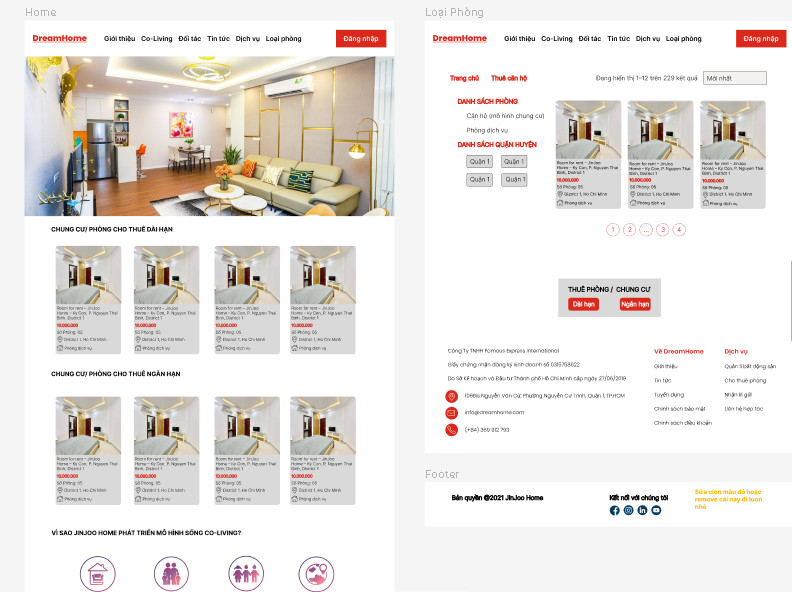
* 1. Site Map

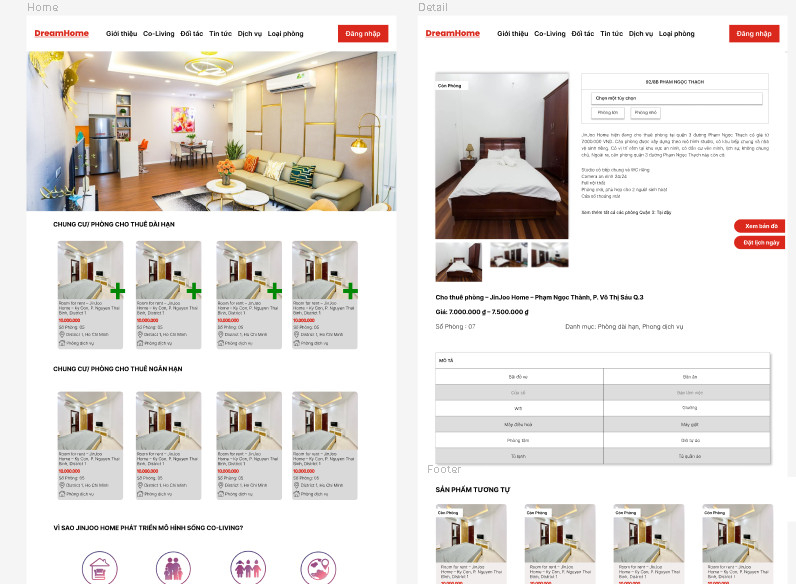
Figure 1 - Site map of the project

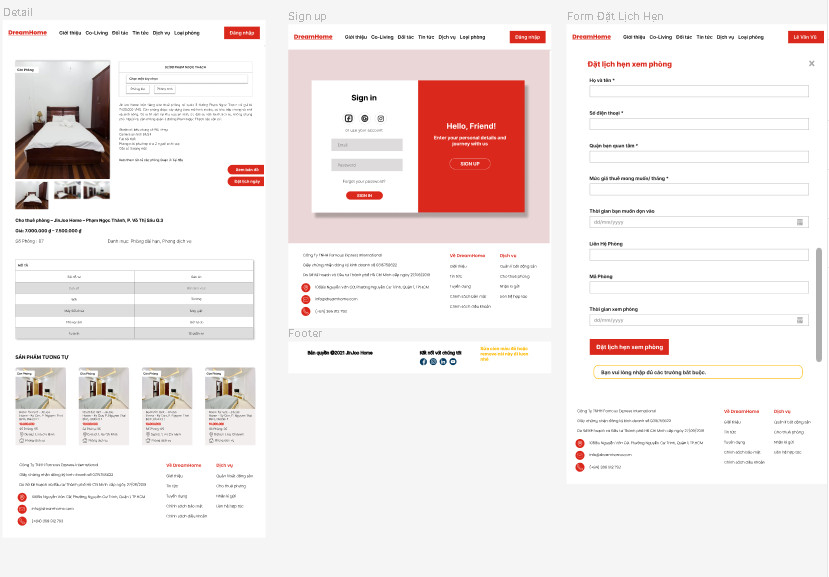
* 1. Prototype designed with Figma









 *Figure 2 - Prototype with Figma*

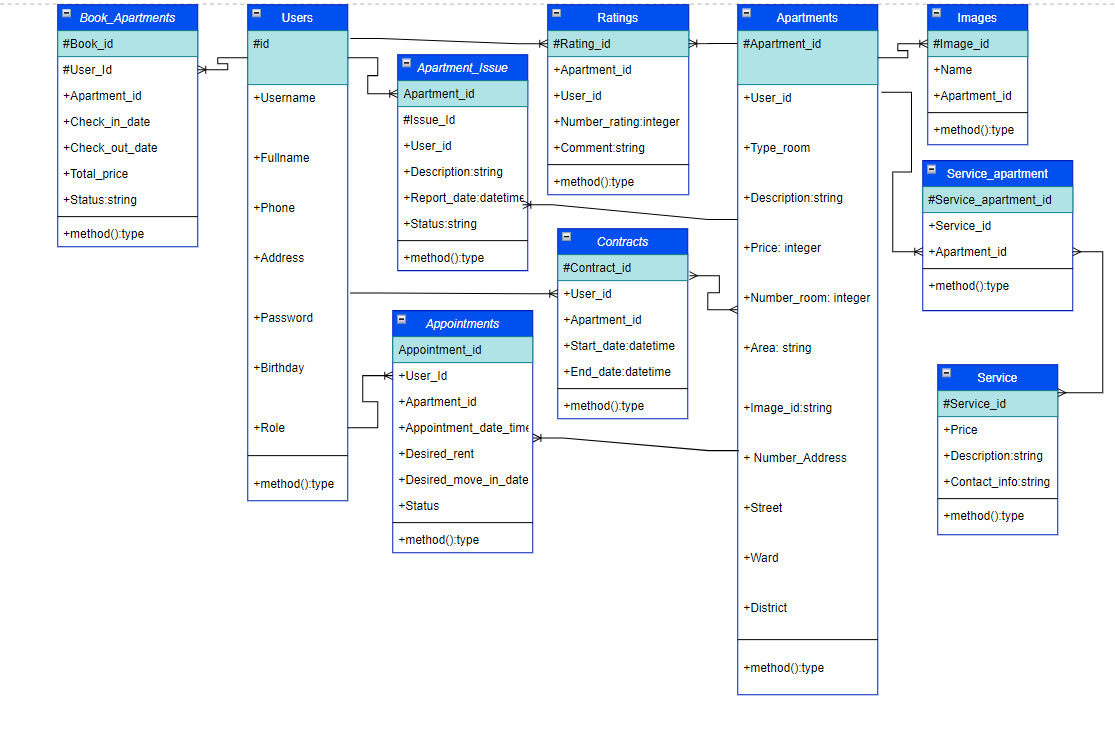
* 1. Entity Relationship Diagram (ERD)

Figure 3 - ERD of the project

* 1. API endpoints

Table 2 - API endpoints of the project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Route** | **Description** | **Params** | **Return** |
| **GET** | /get-apartment | Show all data of the apartment | Apartment Id | return all apartments |
| **GET** | /get-apartment/{apartment ID} | Get the apartment id to specify to certain record | getOneApartment id | Returns an apartment |
| **POST** | /add-apartment | A new object is created from the Apartments class. This object represents a new apartment that will be added to the database | addApartments | Add an apartment |
| **DELETE** | /delete apartment/{apartment\_id} | Get the apartment id to specify to delete a certain apartment | deleteApartments | delete a specified apartment |
| **PUT** | /edit apartment/{apartment\_id} | Push the corrected apartment information to that apartment id | editApartments | edit a specified apartment |
| **GET** | /get-address | Get address of each apartment | getAddresses | return the address of the apartment |
| **GET** | /get-address/{address ID} | Find id address to get only address | getOneAddresses | Returns the specific address of an apartment |
| **POST** | /add-adress | Add an address | addAddresses | add a new address for the apartment |
| **DELETE** | /delete address/{address\_id} | Get the address id to specify to delete a certain apartment | deleteAddresses | delete a specified address |
| **PUT** | /edit address/{adress\_id} | Get the id of the address to correct the address and put the address information back up | editAddresses | edit a specified address |
| **GET** | /get-user | Get User's Information | getUser | return all user information |
| **GET** | /get-user/{id} | Get the user's id | getOneUsers | return specified user information |
| **DELETE** | /delete-user/{id} | Get user id to delete specified user | deleteUsers | edit a specified user |
| **POST** | login | Push login information | login | If the information is correct, the login is successful |
|  |  |  |  |  |
| **POST** | register | Push register information | register | Get information and make registration |
| **GROUP** | middleware | If it accepts the login then it will execute the middleware | api | There will be steps to logout and refresh upon successful login |
| **POST** | logout | Push logout information | logout | successful logout |
| **POST** | Refresh | Push tokens out | Refresh | Refresh token if old token expires |
| **POST** | me | Get the user's information out when logging in, registering | me | Returns current user information after authentication |
| **POST** | /bookings | Post request to create a new booking for an apartment. | store | reservation information is stored |
| **POST** | /bookApointment | Make an appointment | store | store long-term apartment rental appointment information. |
| **GET** | /photos/{apartment\_id} | get a list of images related to the apartment | getRelatedPhotos |  |
| **POST** | add-photo/{apartment\_id} | handle POST request to add a new image to the apartment | addPhoto | If the conditions are right, add a successful photo to the apartment |
| **POST** | /upload | get a list of photos related to an apartment | upload | returns a list of photos related to a particular apartment. |
| **GET** | /get-appointment | Get a list of all apartments | getApartments | return all apartments |
| **GET** | /get-appoinment/{appointment} | Get information of a specific apartment | getOneApartment |  |
| **POST** | /add-appointment | Add a new apartment | addApartments |  |
| **DELETE** | /delete appointment/{appointment\_id} | Delete a specific apartment identified by 'id' | deleteApartments |  |
| **PUT** | /edit appointment/{appointment\_id} | Edit information of a specific apartment | editApartments |  |
| **GET** | /get-confirmappointment | Get a list of all appointments and corresponding user information | getAppointment | return appointment information |
| **GET** | /get-confirmappointment/{appointment\_id} | Get a list of all appointments and the corresponding user information Get the information of a specific appointment identified by $id | getOneAppoiment | returns information about a specific appointment |
| **PUT** | update-appointment/{id} | Update the status of a specific appointment | update | returns the status of an appointment |
| **POST** | /ratings | store new rating in database | store | Returns the user's rating information about an apartment. |
| **GET** | /ratings | get the number of stars of the rating for a particular apartment | getRatings | Returns the number of star reviews for an apartment |
| **GET** | /rating-count/{id} | get all the reviews of a particular apartment | getRatingCountByUser | Returns a list of reviews for an apartment |
| **GET** | /get-service | Show all information about the service |  |  |

**4/ Conclusion**

In this conclusion, we will summarize the DreamHomeFinder project, the results achieved and the learning experiences in the implementation process.

**4.1: Summary**

The DreamHomeFinder project has accomplished its goal of building a web application that allows users to search and post rentals of apartments and houses. The application provides an intuitive, easy to use and responsive interface so that users can easily find information about the apartments.

**4.2: Features and technology**

DreamHomeFinder integrates two main technologies, React and Laravel. React is used for the frontend, providing a modern and interactive user interface. Laravel is used as a backend framework to handle business logic, manage databases, and provide APIs for the frontend.

The main features of DreamHomeFinder include:

- User login and registration.

- Search and filter apartments and houses based on criteria such as location, area, price, room type (short-term and long-term).

- View detailed information about each apartment, including photos, description, amenities, location, and more.

- Post apartments for rent and manage individual apartments.

- Interact with other users through comments and reviews.

- User management, including updating personal information.

- Management of apartment rental contracts.

**4.3: Results and achievements**

During the implementation of DreamHomeFinder, we have achieved the following results and achievements:

- Build an attractive, friendly interface, making it easy for users to interact with the application.

- Implement a powerful search system, allowing users to filter and search for apartments based on specific criteria.

- Integrating the function of posting apartment rentals and managing individual apartments, helping people to easily monitor their apartment management.

- Build the function of commenting, rating, creating opportunities for users.

- Building a user management system, information security and providing individual account management functions.

**4.4: Difficulties and learning**

During the implementation of the project, we encountered some difficulties and learned from them:

- Integration between React and Laravel requires a solid knowledge of both technologies, and understanding and coordinating between them takes time and effort.

- Database management and query optimization is a challenge, especially when the application grows and has a large number of users.

- Ensuring data security and avoiding security holes is an important issue that needs to be carefully considered and implemented

However, through the process of overcoming these difficulties, we have learned valuable skills and experiences, including:

- Frontend development skills with React and effective UI design.

- Skills in API building and backend management with Laravel.

- Database management and query optimization skills.

- Knowledge of web application security and vulnerability prevention methods.

**4.5: Evaluation and next development direction**

The DreamHomeFinder project has achieved remarkable results and brought value to users. However, there are still some improvements and further developments that can be made:

- Expand and improve the search function to meet more complex search requirements and criteria.

- Integrate additional features such as viewing geographic maps, financial advice.

- Improve performance and optimize the application to better respond to large volumes of traffic and users.

- Develop mobile apps for DreamHomeFinder to provide a more flexible user experience on mobile phones and tablets.

**Conclude**

DreamHomeFinder has been a successful project that combines React and Laravel to build an efficient web application for apartment search and listing. Through the implementation process, we have learned and developed many new skills and knowledge. With further improvements and developments, DreamHomeFinder could become a popular and useful application in both business and real estate.