HollidayAcc: CIAI Project Report

Bruno Carmo 57418
Sahil Kumar 57449

December 19, 2023

1 User Stories and Mock-ups

Our client-side single-page (web) application implements the following user-stories:

1. As a client, I want to see the first page of a list of apartments so that I can choose one to rent and see its details.
2. As a client, I want to see the next page in the list of apartments so that I can choose one to rent and see its details.
3. As a client, I want to see the details of an apartment so that I can read its description and amenities and decide if I want to rent it.
4. As a client, I want to see the calendar of an apartment so that I can decide when to rent it.
5. As a client, I want to see the reviews of an apartment so that I can decide if I want to rent it.
6. As a client, I want to select a starting date and an ending date in the calendar of an apartment to start making a reservation.
7. As a client, I want to fill a form with the number of people to finish making a reservation.
8. As a client, I want to see the details of a reservation so that I can see if it was accepted.
9. As a client, I want to see the details of a reservation so that I can cancel it and see the result in the details of the reservation.
10. As a client, I want to see the details of a reservation so that I can check in and see the result in the details of the reservation.
11. As a client, I want to see the details of a reservation so that I can check out and see the result in the the details of the reservation.
12. As an owner, I want to see the list of my apartment so that I can select one apartment and see its details.
13. As an owner, I want to see the details of an apartment so that I can add a period of availability and see the result in the calendar.
14. As an owner, I want to see the details of an apartment so that I can see the list of reservations for it.
15. As an owner, I want to see the details of a reservation so that I can accept it and see the result in the list of reservations.

To better visualize the overall feel and navigation between the pages we started by creating some mock-ups (presented in figure 1) for each of the user stories. This also helped a lot in creating the IFML specification and the final product.

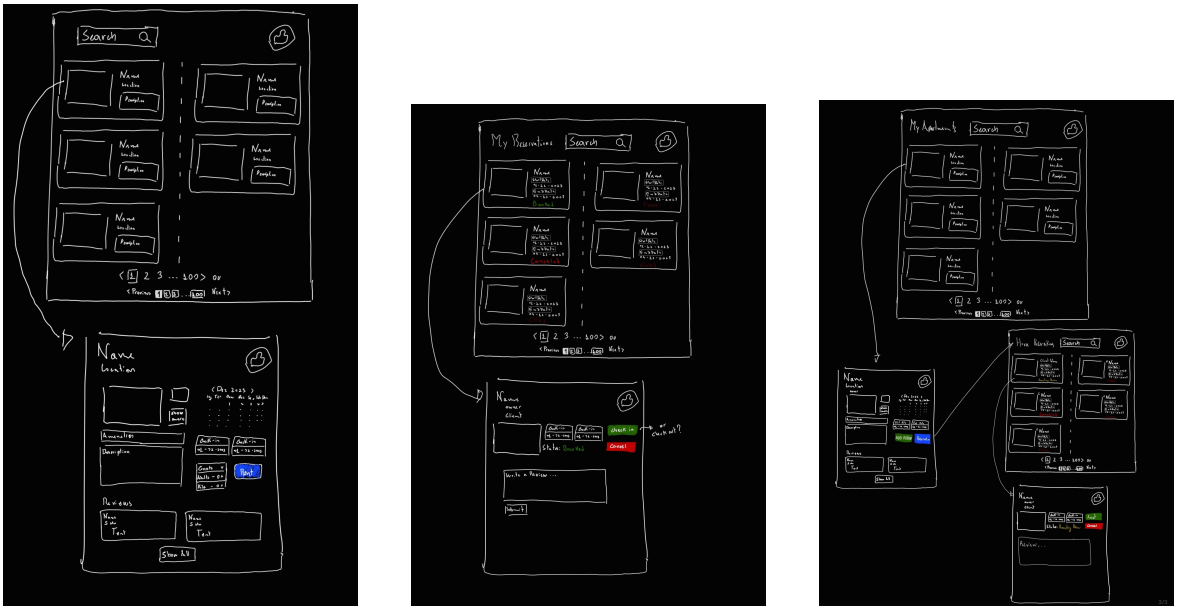

Figure 1: Initial mock-up of the client app

2 IFML

IFML defines the behavior and helps define the components of UI parts. It defines the flow of the user between the various screens of the application and the data transmitted between the different screens. The IFML diagram for our application is presented in figure 2.

Our homepage is defined as an XOR because the user can navigate to all these pages (Apartments, Owner Apartments, User Reservations, User Details) from any of these pages. This is implemented via a navigation bar at the top of the screen where the user, upon logging in, can check their resources such as their owned apartments, reservations, and user information.

In any page containing a list we have dedicated search and paging actions so that the user can always filter through the various results, may they be apartments or reservations.

In our application, we define our landing page as being the main focus, the apartment searching. From here a user can select a particular apartment and is redirected to the apartment details page. Here the user can observe the apartment details, its periods, and reservations on a calendar. The user can then click on the check-in and exit dates on the calendar, choose the amount of guests, and then rent the apartment.

A user can then check their reservations in the navigation bar, select the reservation, and check its details. In this menu, the user can check in, check out, or review the apartment depending on its status.

If the user owns apartments, he can check their apartment list in the navigation bar, choose one, and see its details. Here the user can add a new period by selecting a start date and end date. Here the user can navigate to the house's reservation list. Upon selecting a reservation here, the user can check the details of the reservation and can accept the reservation or cancel it, depending on the status of the reservation.

3 ChatGPT

We did not use ChatGPT nor GitHub Copilot nor any other LLM tool in our project.

4 Major Challenges

One of the major challenges we had in implementing our front-end application was the implementation of a react calendar component. For this, we found an interesting booking calendar component for react

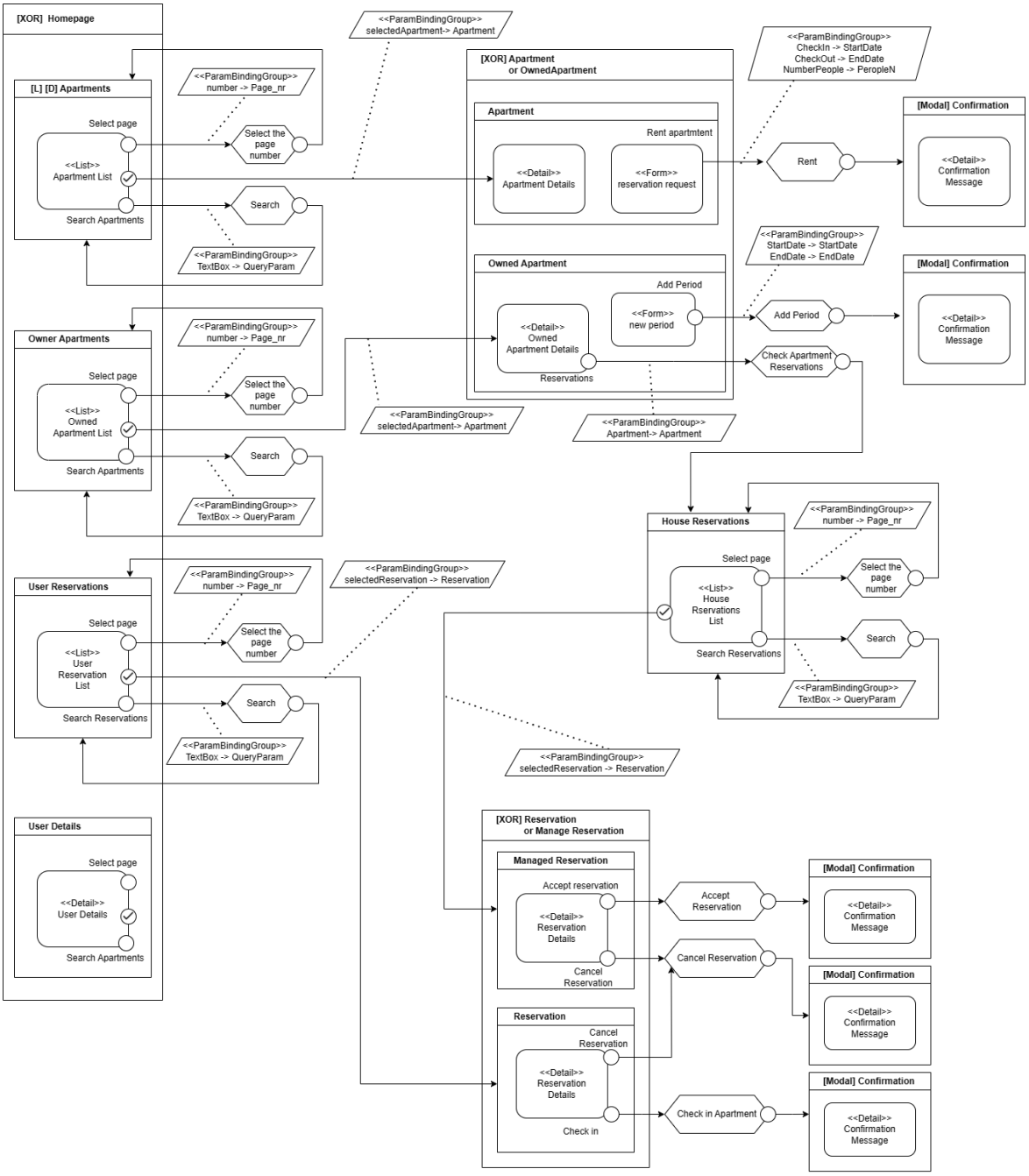

Figure 2: IFML Diagram

[1] but upon trying it out we had many bugs and issues with it. We solved this problem by downgrading the version of the package, but due to the requisites for this project we needed to change how the calendar worked (there was no concept of date availability, its purpose was only for booking) so we downloaded the source code, analyzed it, and modified it for our needs.

Another major challenge we encountered was outdated information when developing the backend of our application. Following our database schema we needed to use composite primary keys. Spring JPA offers two ways to implement it: *EmbeddedId* annotation and *IdClass* annotation. Considering that using *IdClass* was simpler than using *EmbeddedId* we followed that approach. However, we later found out that *IdClass* does not work correctly for some versions of Spring JPA and unfortunately, ours was one of those versions.

[1] https://www.npmjs.com/package/@demark-pro/react-booking-calendar

Lastly, a big challenge we confronted was debugging our code considering we used a big framework like Spring. For example, we formulated the states of a reservation as a list of states where the first element of that state is the current state of the reservation. As such, we always inserted a new state at the start of the list. However, for some unknown reason to us, even though, we inserted a new state at the start of the list, when conforming to where the state was inserted, it was at the end of the list. Another debugging problem that led to our first delivery being without basic authentication was that for some unknown reason to us when defining the API method calls, the method should have a parameter of type *Principal* representing the authenticated user. We send that parameter to our *Security Service* to check if the user can make the API method call (e.g. only the owner of a house is permitted to add periods to it). However, in the definition of the method in our *Security Service* the principal parameter should be of type *User*. Knowing only this second part, we defined our API methods with a parameter of type *User* representing the authenticated user, leading to a lot of wasted time.

5 Extra Stuff

Finally, because it was mentioned in the lectures we made use of the Bootstrap framework to help us design our application. We used various components such as the NavBar, Input, Buttons, Cards and the Grid system with its rows and columns to build the layout for the application.

We used various CSS classes and animations for some of our components such as the cards for the lists of houses, reservations, and reviews.

We implemented a login feature in the navigation bar where the user can easily enter an account or change accounts. We fully added the review aspect to the application, where the user can review the apartment after finishing a reservation, giving it a review description and rating. This review is then added to the details of the apartment.