Discussion

Here we present several issues that we encountered during the development of this application along with design and implementation decisions, technologies used, and possible improvements and extensions to our take on QBnB.

Problems encountered during development and solutions

Version control

One of the more prominent issues we had during development was version control. We had a team of three people and it was often difficult to manage versions of the project. Often we would have several different versions of the project ongoing at the same time with different people going in different directions. Our solution to this was to use GitHub to track for versions. There was a learning curve involved but once there, it was certainly easier to control version and manage changes.

Program architecture and workflow

Initially, we tried to use the Model View Controller architecture for our program. We know from class and online that this is one of the more prominent frameworks for web development. Unfortunately, we knew very little about it and were short on time so our solution was to use a basic object-oriented framework. For instance, we had a class that managed all our database transactions, a class to model each user and the things they can do, a class to manage different properties, etc. This worked well as it allowed us to separate our work: one of us could design the object abstraction, someone could work on the UX design in HTML, and someone could work on the documentation/organization of the code.

Old and deprecated PHP functions

When trying to learn all of the PHP and HTML required to create this application, we went to several tutorials on Moodle and online to learn the basics. Unfortunately, we ran into a lot of issues using old (and unsafe) functions and coding practices that we didn't know were actually insecure. This resulted us going over a lot of our code near the end to ensure that SQL queries were safely executed and connection objects were always closed, etc.

Learning PHP's session-oriented features

In order to keep track of who was logged in and which property was being viewed at any point in time we took advantage of PHP's 'SESSION' objects. This allowed the user to open a session with the server, perform any action he/she wanted to do, concurrently with other users; however, initially we were unfamiliar with the capabilities of the session and post variables. The main issue we encountered was a conflict when the admin signed in and viewed information about other members. When the admin signed in, we set the session's member id to the admin's. This resulted in some confusion of who was actually signed in and if he/she was an admin or not. We fixed this by checking for the admin rights early in the application flow.

Design and implementation decisions

Comments

We allowed each member to make a comment on any property because they could ask questions about properties that they were interested in booking. As well, they could reply to any comments someone else made to add into the conversation.

Dates Booking Conflicts

Using the assumption stated in the project outline that no bookings would conflict. The owner was the person who rejected the booking if they conflicted.

Technologies used

GitHub

We used the desktop version of GitHub to develop the QBnB website. This allowed our team to make sure that each update was compatible with what other people were doing. It allowed us to work productively while not physically beside each other. As well, we were able to monitor what we have done on the database thus far and keep track of everything. The repository we used was public and so it would have been better if it was private. Big learning curve.

XAMPP

We used XAMPP as our server to connect our PHP script to the database.

PHP

We used PHP to connect our web application to the database. This allowed us to dynamically show pages in our application. We created classes to keep track of members, properties and the database. PHP was used at the recommendation done during class. Since this was the first time any of our team has used PHP we didn’t exploit many of its capabilities, next time we will.

MySQL

We used MySQL to connect to the database in Apache. This was done because we were familiar of how it worked through class. We were implementing fairly rigorous SQL but due to the incremental parts of the project, in the last phase we didn’t create many new SQL statements.

Possible improvements and extensions

Application architecture and framework

Looking back, our framework for the program was minimal or non-existent. Despite having a decent object-oriented model, there was still clear interdependency between the workflow of all the members. There were often pauses in development and the general file organization was poor. Adapting the Model-View-Control architecture would have solved a lot of these issues.

Implementation on a web server

When we were accessing our server it was strictly on our local computers. To actually implement QBnB, we would certainly need to purchase a host server to store all our files and get a website domain like www.qbnb.com.

Bootstrap

Our application doesn't have any CSS implemented. If we were to implement a CSS wrapper in our web application, such as Bootstrap, it would make the user experience much better. Unfortunately, due to time constraints, we struggled with extending this part of the assignment.

Geo-locating in the application

Currently members have to enter the district that they are in. An extension to the project is that they type the address and then the district can be filled in automatically. This would need intergration with some sort of apps.

Payment

Currently QBnB doesn't support payments; integrating something like stripe into our website would allow people to make payments with a turnkey system. We assume in our system that the payment will occur in person when the tenant goes to the accommodation.

Password Encryption/Hashing

For this iteration we didn't hash the password. In order to create a better security for our application, we would need to encrypt and hash the password and then store the password instead.