COSC2471

iPhone Software Engineering

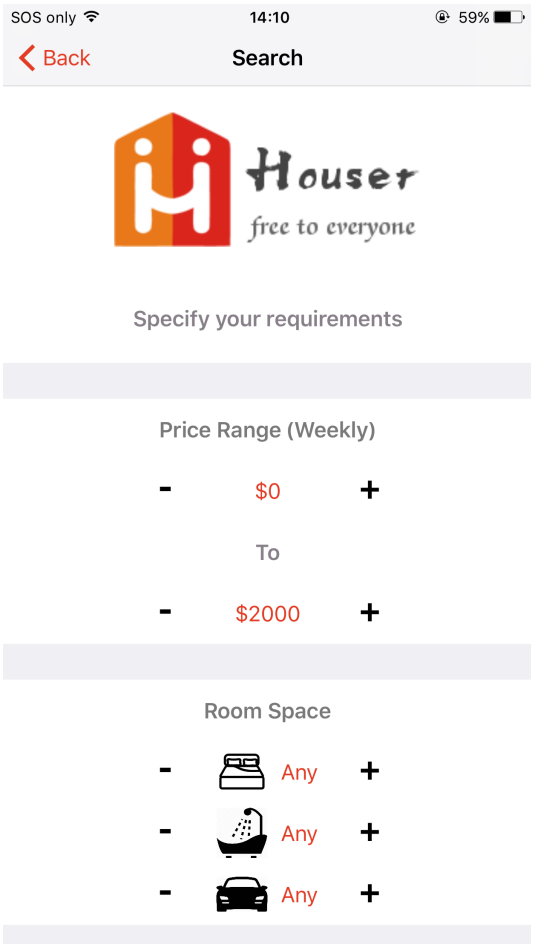
Assignment 2

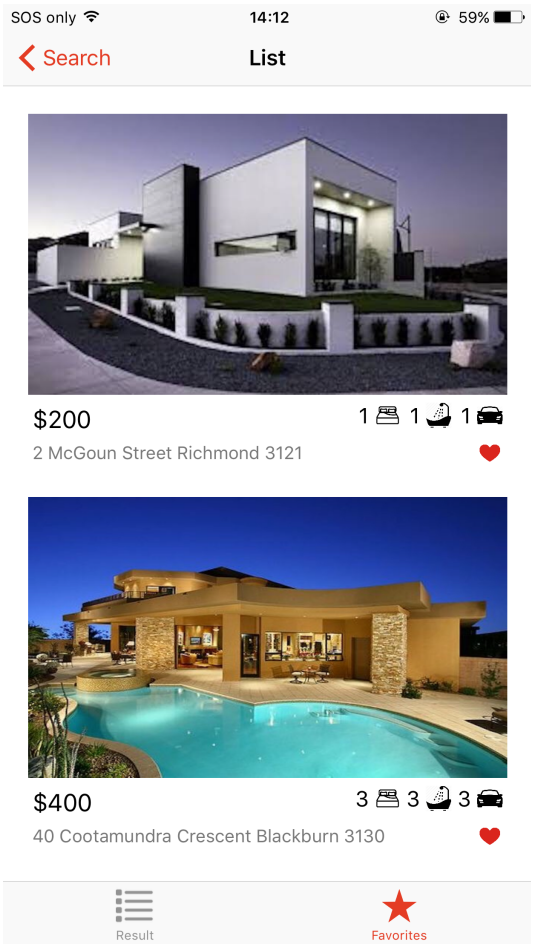
Ning Nan

nanning9925@gmail.com

Table of Concepts

Design Principles





Deference

- Contents fill then entire screen.
- Contents and tasks differ from each other with different RGB color.
- No need for user to login.
- Minimal use of bezels, gradients and drop shadows. Interface is light and airy.
- Each Scenes focus on one task.

Clarity

- Labels and texts show clearly different layers. Larger front size shows more important information.
- Icons are presenting their own meaning.
- Buttons such as full red heart icon and the border only red heart icon conveys interactivity.
- Space is never crowded.

Depth

- Use of popover reveals part of the screen to remain users when doing the sensitive operation.
- Icons on the pictures can be clickable to mark or unmark one property.

Contrast

- Most of the clickable elements in the application is using the same color which comes from the logo color. And these colors make contrast with the labels and other elements. Gives a good color feel and clearly reminds users what operations can be done next.
- Labels of the results details make contrast with each other, price label will be bigger as most people consider about price first.

Repetition

- For all the lists, Result, Favorite and My Property List, the properties will be displayed in cells. And cells share the same layout and icons.
- Users do not need to learn much as senses are often using the same design.
- Colors of buttons are all using same color.

Alignment

- For all the cells that display properties. Labels are fixed position in every cell. And align with edges or each other. If users only focus on one attribute of the property, they can simply focus on one point on the screen and scroll down.

Proximity

- For all the cells, attributes belong to the same property will always be tightened together. And different cells have different contents.

Design Patterns

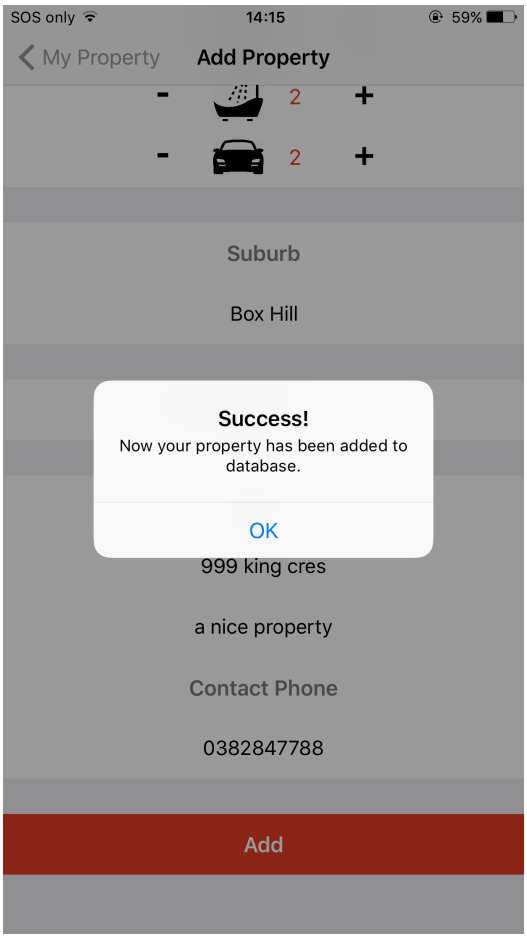
Delegation

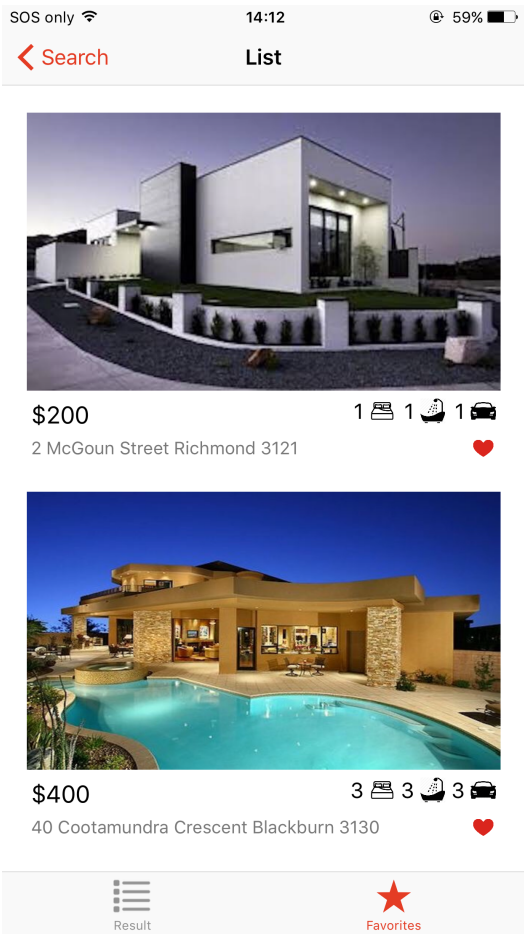
```
protocol DatabaseOperations{

    func queryDatabase(operation: String) -> FMResultSet?
    func modifyDatabase (operation: String) -> Bool
}

class DatabaseOperation : DatabaseOperations{


    func queryDatabase(operation: String) -> FMResultSet?{


        if(!database.open()){

            print("Error: \(database.lastErrorMessage())")

            database.close()
            return nil
        }
        else{

            let rs = database.executeQuery(operation, withArgumentsIn: [])


            return rs
        }
    }

    func modifyDatabase (operation: String) -> Bool{

        if(!database.open()){

            print("Error: \(database.lastErrorMessage())")

            database.close()
            return false
        }
        else{

            let rs = database.executeUpdate(operation, withArgumentsIn: [])
            database.close()
            return rs
        }


    }
```

- Protocol and Class as above. DatabaseOperation Class contains the basic database operations : query and modify.

- Then one another class Model class use delegation to use the basic database operation in the above class. All of the functions that related to database has used delegation.

CRUD Operations





Create

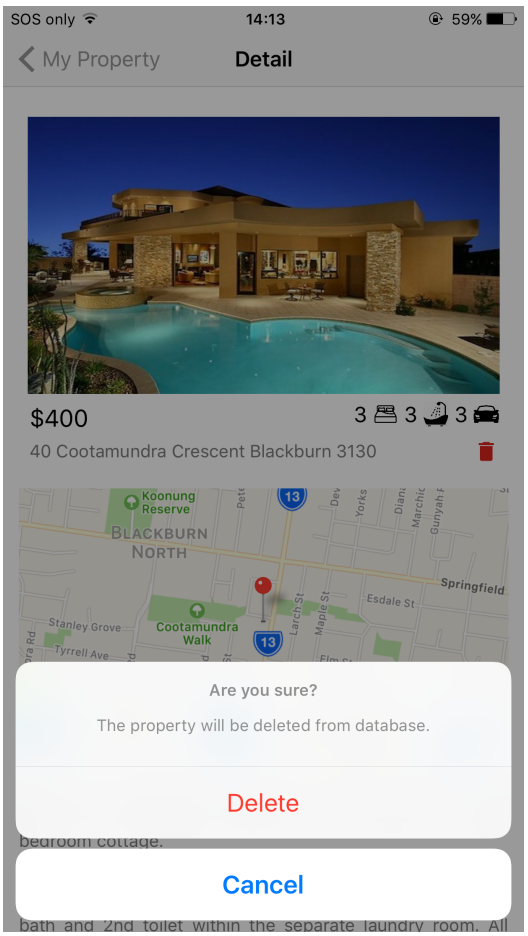
- In the landlord page, while user click to add property, and fill the blanks with no empty messages and valid information then click Add. The application will check these information first. Then the data will be added into the database by creating one row.

Read

- For all the lists(Result, favorite and owning), before the table load, the application will first read from database. Then display in the cells. The read is automatically every time before the table loads.

Update

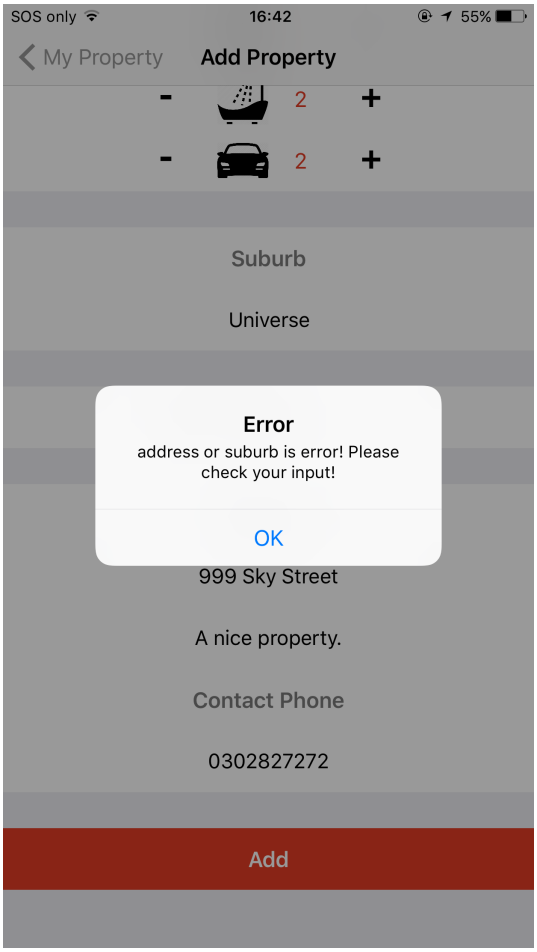
- The red heart icon just below the image shows the favorite status in the database. If users click on the icon, the icon will change to border only and update the favorite status in the database.

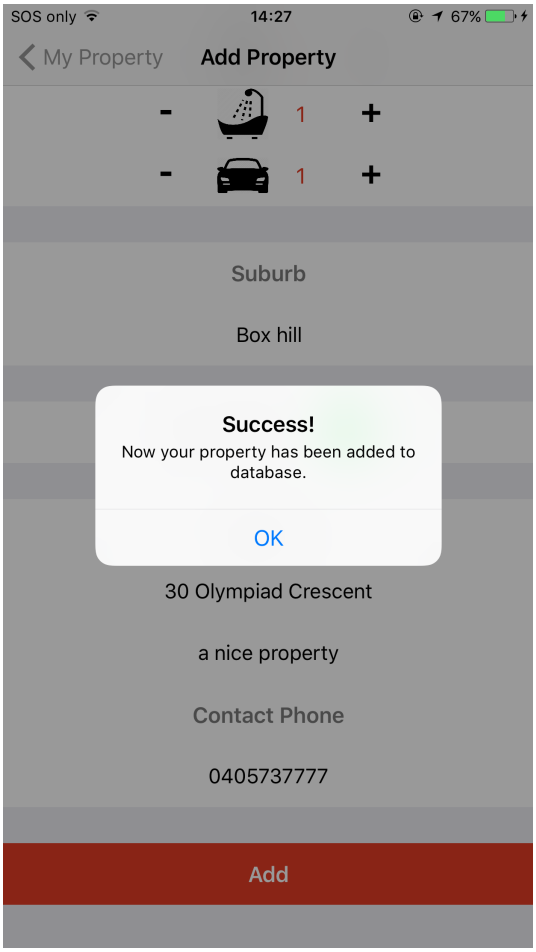


Delete

- While user enters the detail page as landlord, then he/she can see the favorite icon becomes delete button. If user clicks on the button, the action sheet will present. If user continue to delete, then the row of this property in the database will be deleted.

REST Implementation





REST API example request:
https://maps.googleapis.com/maps/api/geocode/json?address=30 Olympiad Cres, Box Hill North&components=administrative_area:VIC|country:AU&key=AIzaSyD8X2n5jN8x7Rddir7FFglz0laMvNCK9fs

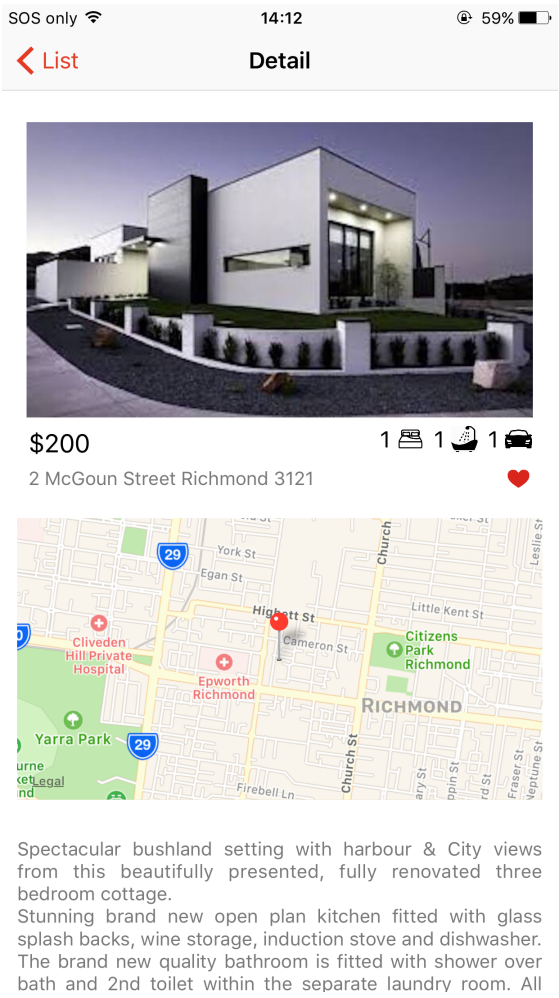
Input street detail with suburb. And the limitation of VIC and AU.
Return address detail, post code, coordinate and many other information about this address.

Here I will extract the post code and coordinate. If the address and suburb are not valid in the VIC. The request will return error, thus no post code and no coordinate, then the application will give error message.

If the address and suburb are valid. The coordinate and post code will be added into the database for other use.

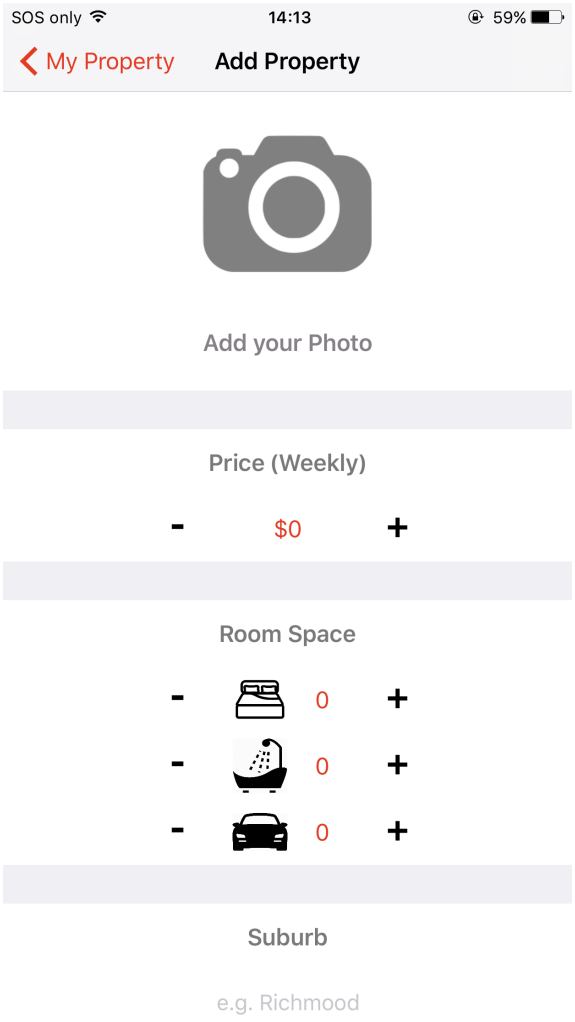
So, my REST API Implementation is used to check the address(Not too accurate, but still effective), get post code and get coordinate(used for next chapter, map).

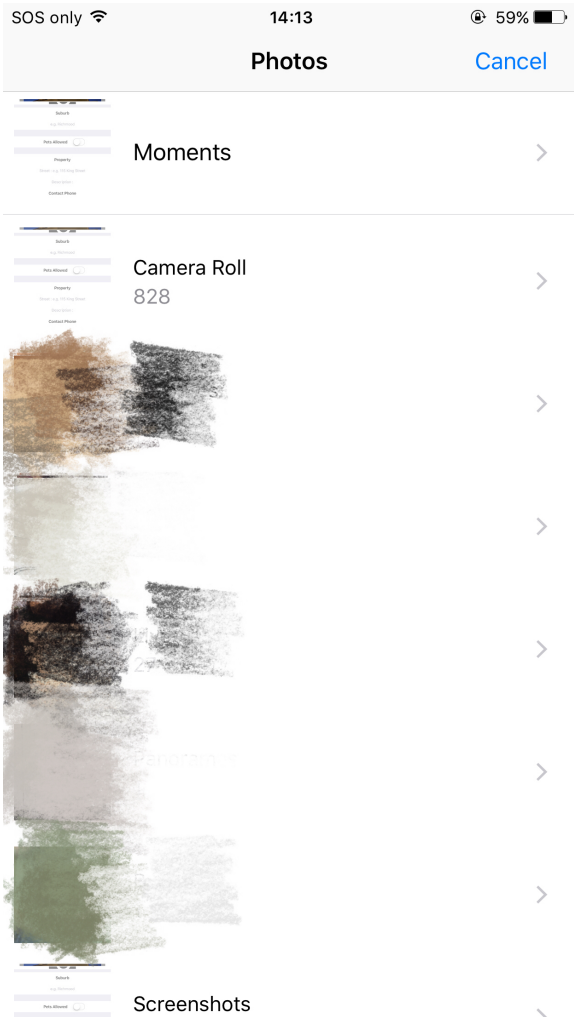
Map



As I mentioned last chapter, through the REST API, now we have the coordinate for the property. When the user enters the detail page, the detail page will load the coordinate from the database. And then use coordinate into the core map. Then user is able to see where the property is.

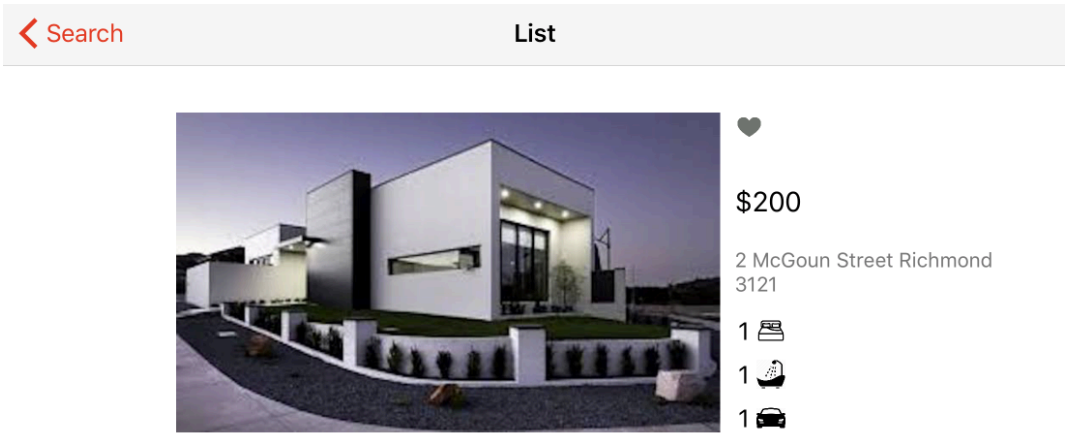
Photo

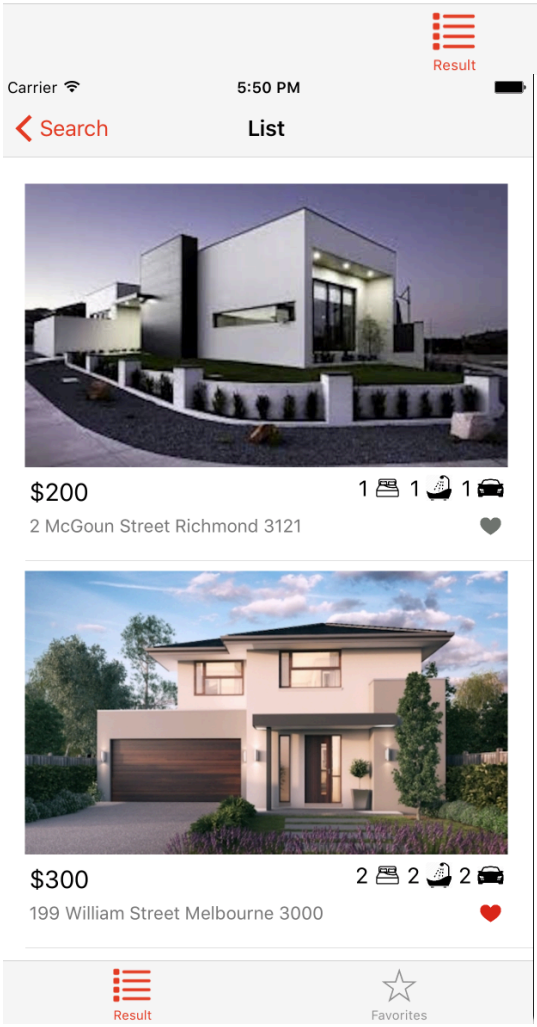




User is able to upload photos for the property. When user enters the Add Property page, they can click the photo image to choose the photo. After the photo is chosen, the photo will be displayed in the add page. And after the add operation is successful, the photo will be saved in the documents.

Size Classes





- While the device is the size (w: R h: C) (iPhone 7 Plus), the icons, texts will be displayed on the right of the image.

- The normal situation is, icons, texts will be under the image.

References

FMDatabase: https://ccgus.github.io/fmdb/