**iFlat**

System Design

Version 1.0

11.12.2016

Alican Yılmaz  
Tolga Taner  
Eren AY

Ö. Meva Demiröz

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc433996772)

[1.1. Purpose of the System 1](#_Toc433996773)

[1.2. Design Goals 2](#_Toc433996774)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc433996775)

[1.4. References 3](#_Toc433996776)

[2. Current Software Architecture 4](#_Toc433996777)

[3. Proposed Software Architecture 5](#_Toc433996778)

[3.1. Overview 5](#_Toc433996779)

[3.2. System Decomposition 5](#_Toc433996780)

[3.3. Hardware Software Mapping 6](#_Toc433996781)

[3.4. Persistent Data Management 7](#_Toc433996782)

[3.5. Access Control and Security 8](#_Toc433996783)

[3.6. Global Software Control 9](#_Toc433996784)

[3.7. Boundary Conditions 10](#_Toc433996785)

[4. Subsystem Services 11](#_Toc433996786)

[5. References 12](#_Toc433996787)

SYSTEM DESIGN DOCUMENT[1]

The System Design Document (SDD) is written after the initial system decomposition is done, and updated throughout the development. SDD describes the services provided by each subsystem. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

SDD is used to define interfaces between teams of developers and serve as a reference when architecture-level decisions need to be revisited. The audience for the SDD includes the project management, the system architects (i.e., the developers who participate in the system design), and the developers who design and implement each subsystem.

# Introduction

The following section provides an overview of the derived System Design Document for the project Flat renting system. Purpose of the system, design goals, definitions, acronyms and abbreviations are specified below. This document also includes current and proposed software architectures, subsystem services and references.

## Purpose of the System

The purpose of the system monetizing extra space of customers and user who is looking for people for renting their space. Our system’s name is iFlat. So rest of the document iFlat is states for our Flat reservation and renting system. Our system is providing a good user experience and usability which requires a few tapping gestures to complete system functions. For renters, creating profiles, editing profiles, rating flats and hosts, registering and logging in, rating flats and hosts, renting flats of hosts and make reservations, cancelling reservations. For hosts who is added any flat on the system, registering and logging in, rating renters, creating host profiles and flat profiles and so on. Our system also let any renter to became host by adding any flat anytime. And users who has flats already on showcase and system also can rent flats.

## Design Goals

The goal of the system is to satisfy the functional and non functional requirements as specified in the requirements specification document. (RAD). Our system provides correct, robust, flexible, reusable, efficient, reliable and usable functionalities. Our system is a mobile application which is on iOS platform. It is consisting of, three side, Admin side which can review the feedback and issues and close/answer them, Renter side, which can reserve flats, creating profile and creating feedbacks and issues and rates flats, the host side is let hosts to create flat profiles, rate renters and creating issues and feedbacks for system to Admin. Hosts are also can reserve flats also. All users should register to our system to make any reservations. This parts are designed user friendly and using all mobile device capabilities like sensors, gestures and more. Our design is readable and content-first apporached, and easy usability.

## Definitions, Acronyms, and Abbreviations

* Mobil Platform => ios [1], [2]
* Databese => FireBase (no sql) [3], [4], [5], [6]
* iFlat => Rent a flat system
* Xcode => Editor [7], [8]
* Swift 3.0 => High level programing language [9], [10], [11], [12]
* JSON => Java script Object Notation [13], [14], [15], [16], [17]
* XML => stands for eXtensible Markup Language. [18], [19]
* MVC => Model-View-Control. [20], [21], [22], [23], [24]
* CocoaPods [25], [26]

## References

1. <https://en.wikipedia.org/wiki/IOS>

2. <https://ipsw.me/>

3. <https://firebase.google.com/docs/ios/setup>

4. <https://www.youtube.com/user/Firebase>

5. <https://firebase.google.com/docs/database/>

6.<https://docs.mongodb.com/?_ga=1.218486233.1118911483.1481064979>

7. <https://developer.apple.com/xcode/>

8. <http://stackoverflow.com/questions/tagged/xcode>

9. <https://swift.org/blog/>

10. <http://swift3tutorials.com/>

11. <https://developer.apple.com/swift/>

12. <https://www.raywenderlich.com/category/swift>

13. <http://www.w3schools.com/js/js_json_intro.asp>

14.<https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/>

15.<https://firebase.google.com/docs/hosting/full-config#section-firebase-json>

16.<http://stackoverflow.com/questions/39423367/correctly-parsing-json-in-swift-3>

17.https://cocoacasts.com/building-a-weather-application-with-swift-3-decoding-json-data-in-swift-part-1/

18. <http://www.theappguruz.com/blog/xml-parsing-using-nsxmlparse-swift>

19. <http://www.w3schools.com/xml/>

20. <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

21. <https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>

22. <https://cocoacasts.com/mastering-model-view-viewmodel-with-swift-3/>

23. <https://blog.thomasdurand.fr/swift3/ios/2016/08/07/ios-basics-tableview-setup.html>

24. <http://codewithchris.com/how-to-make-iphone-apps-mvc-one-pattern-to-rule-them-all/>

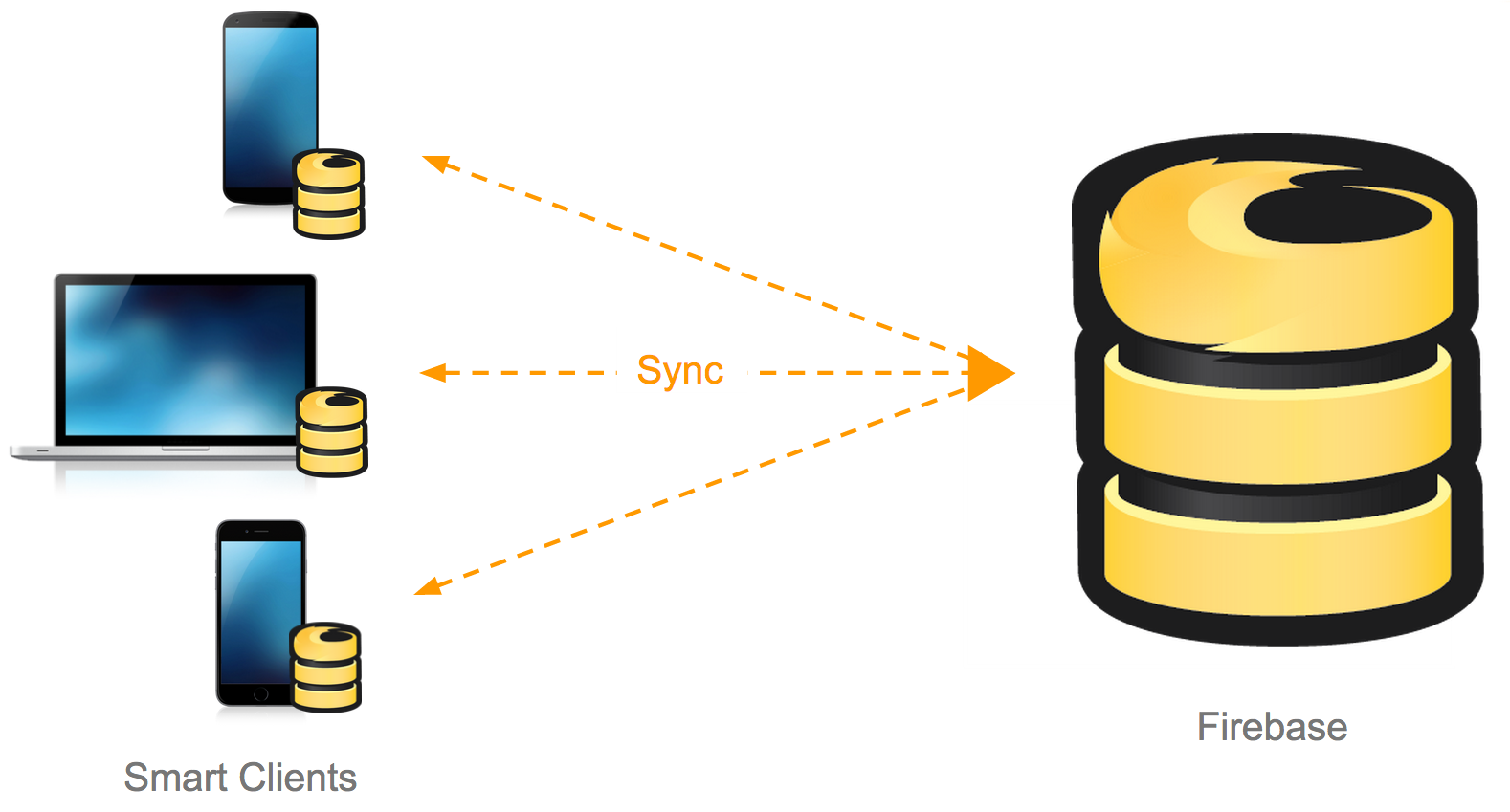
25. <https://cocoapods.org/>

26. https://www.raywenderlich.com/97014/use-cocoapods-with-swift

# Current Software Architecture

The iFlat is divided into layers based on the 2-tier architecture. These are client and firebase in this figure 1. Mobile phone implements the user interface and deals with pages rendering. The firebase SDK requests store all data Our application, the mobil phone and deals with core function alities of the system such as s earch flats, renter a flat, edit profile, edit flats etc.

This architecture will forward the data to client and ensure access to the persistent data like preferences and flats for customer. Client direct connection with firebase also firebase can ensure that only valid data is allowed to update in the firebase.



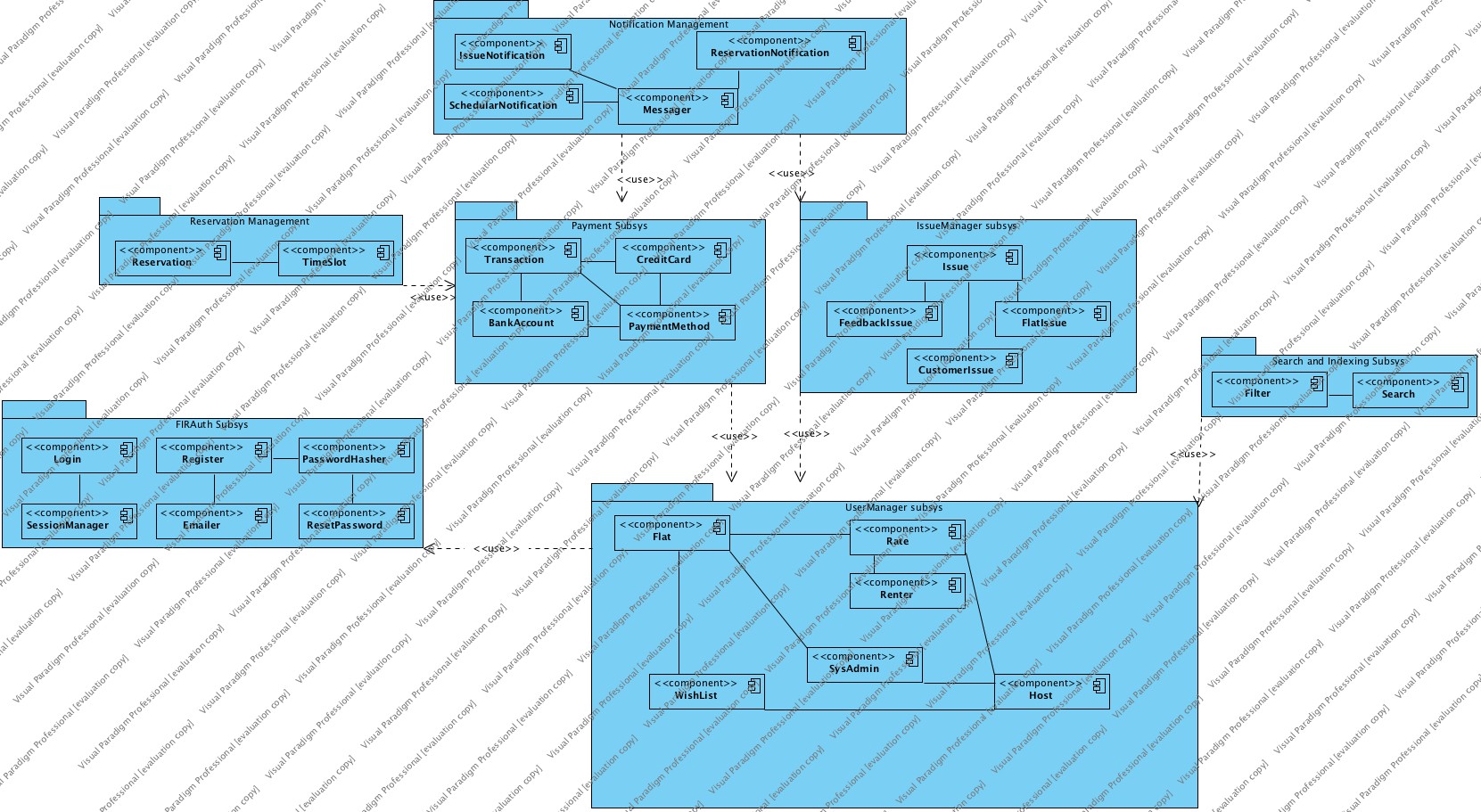
*Figure 1: Firebase BaaS architecture [2]*

# Proposed Software Architecture

## Overview

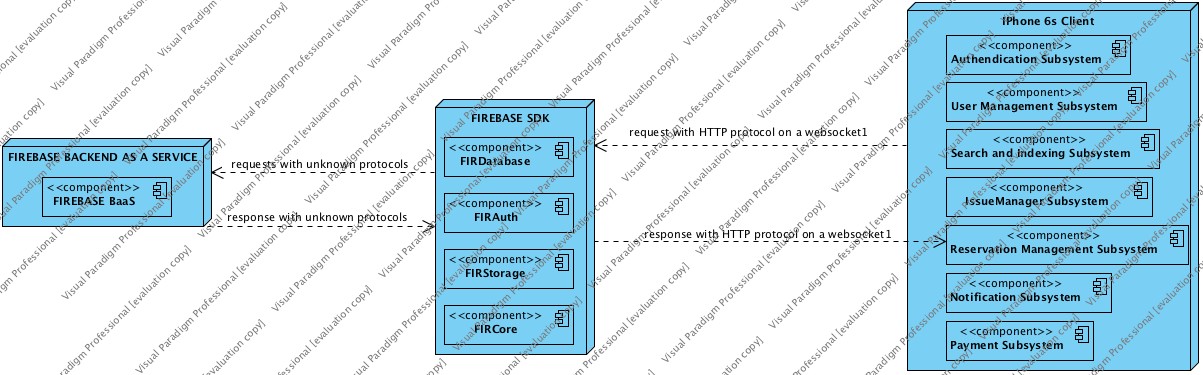
In this system we are using Xcode ide and swift native programming language. Our system using a database motor which adapts nosql apporach. Our system consists of MVC model view controller architectural style. Each view and controllers are designed with team members with associated system package with models and entities.

## System Decomposition



* **Notification Management Subsystem**: This subsystem is responsible to send notification to users when Payment Subsystem is received a successful payment after reservation request. Also Notification Management Subsystem communicator of Issue Manager Subsystem, if any issue has been sent from users, sents notifications to communicators.
* **Reservation Management Subsystem**: This subsystem is responsible to make reservations and creating reservation requests entities. This subsystem is also communicator of payment subsystem, when any payment is occurred, it created reservations according to timeslots of flats.
* **FIRAuth Subsystem**: This subsystem is fully responsible of authendication and implemented by FIREBASE API. This authendication subsystem logs in users, registration, secure credentials, sends emails with communicate with UserManager subsystem.
* **UserManager Subsystem**: This subsystem is responsible with managing entitites information.
* **Search and Indexing Subsys**: This subsystem is responsible with making searches and displaying related entities with communicate with UserManager. Also this subsystem can filter search results.
* **Payment Subsystem:** This subsystem is responsible with making payment transactions and creditcard, bankaccount balance management. This subsyste mis in communicate with user manager subsystem.
* **Issue Maneger Subsys:** This subsystem is responsible with creating issues and handling them. Also this subsystem is responsible with managing Feedbacks, complaints created by users. It delivers related issues, and functions.

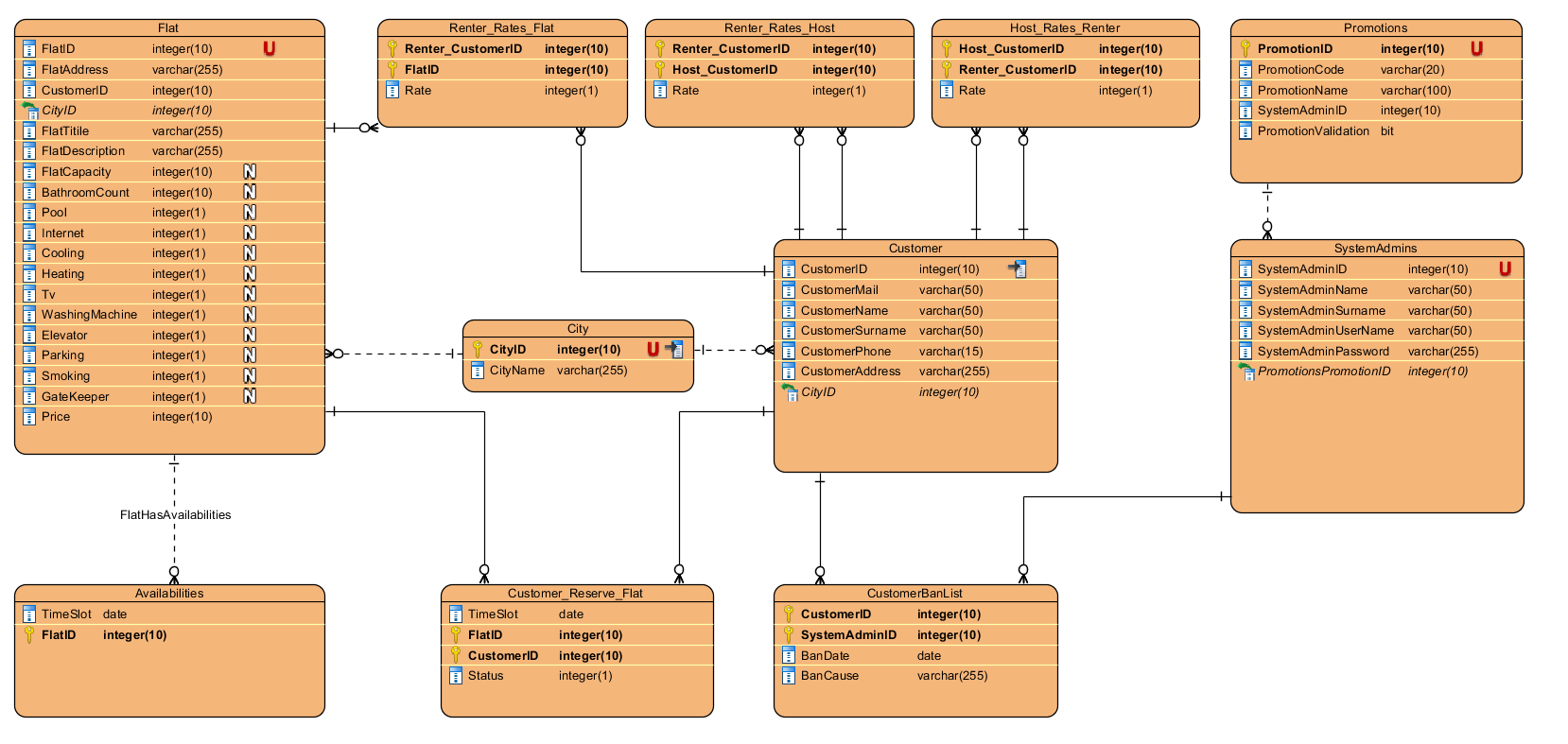
## Hardware Software Mapping



Firabase is a persistent data storage solution which uses No-SQL apporach and stores data as JSON tree with raw text. Firebase is a back-end as a service which is also includes storage and authendication layers with realtime database feature. Firebase is a solution which solves the problem of writing API layer between database server and web server. Firebase omits this layer and provides a SDK which is on Xcode ide which let clients talk directly with servers on sigle websocket. The database is directly sync with clients, so when any data manipulation occurs with any of clients, all clients are notified by Firebase BaaS. This let realtime data sync between all clients. [3]

## Persistent Data Management

We are using a relational model on firebase. But the system has no relation physically. On Json tree. On no-sql apporach, we are denormalizing data and intend to create a flat and shallow datamodel which decreases relations therefore decreasing relations. This requirement is needed for a fast search and fast data manipulation needs. This apporach of course has some cons and pros, but we have followed the below datamodel to create our firebase nosql json datamodel and referenced it.

****

## Access Control and Security

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Assets 🡪  Roles | | Login/  Register | Edit own Profile | Add/Edit  Flat | SearchFlat | Open İssue /CloseIssue | List Flat | ReservaFlat | |
| SysAdmin | | x |  |  |  | x |  |  | |
| Renter | | x | x |  | x | x | x | x | |
| Host | | x | x | x | x | x | x | x | |
| customer | | x | x |  | x | x | x | x | |
| Firebase SDK is working on client-side code therefore firebase’s security is differenf mysql databese. Firebase has 4 different security features that are social media login, email login, single-session anonymous login, custom login tokens like SSO (single-sign on) which is a session and user authentication, Firebase has declarative language for authorization and we write own security rules on the firebase’s security&rules tabs like data validation. [4], [5] | | | | | | | |

## Global Software Control

The iFlat architecture have a decentralized software control. The system's dynamic control is distributed among different controllers such that each object delegates some responsibility to other objects. The request initiations are event-driven:

* The FirAuth subsystem is initiated by opening the iFlat application.
* The Search and Indexing Subsystem initiated when logging into the system.
* The Reservation Subsystem is initiated when search request is made by Search and Indexing subsystem by using filter or not.
* The Reservation Subsystem initiates the Payment Subsystem after successful reservation request by activating an payment api.
* The Payment Subsystem initiates the Notification Subsystem by gathering information from the flat, renter and payment.
* The IssueManager Subsystem initiates the Notification Subsystem by opening an issue or closing an issue.
* The IssueManager Subsystem initiated by the UserManagement Subsystem by creating an issue about user.
* The IssueManager Subsystem initiated by the ReservationManagement Subsystem by creating an issue about flat.
* The UserManager Subsystem is initiated by UserManager Subsystem when user rates user.
* The UserManager Subsystem is initiated by UserManager Subsystem when user rates flat.
* The UserManager Subsystem is initiated by FirAuth Subsystem when user wants to edit or show user profile.
* The UserManager Subsystem is initiated by UserManager Subsystem when user makes a wish.
* The Firebase Subsystem can be initiated at any point by all Subsystems except the UI Subsystem. Any request (add/update/delete/retrieve) to the repository initiates the Firebase Subsystem. The system stores its contents in the firebase between executions. When the application is run again, it retrieves the contents from the previous execution. Any change in the contents during this execution updates the firebase.

## Boundary Conditions

* **Startup:** Customer taps on iFlat application on his/her mobile phone.
* **Shutdown:** Customer logout form the system.
* **Error Conditions:**
  + **Authendication**
    - Wrong credential info.
  + **Register:**
    - Weak password.
    - Not verified email
    - Bad email
    - Bad birthdate
    - No photo
    - Bad Info
  + **Edit User Profile Info:**
    - Bad user info mentioned in register
  + **Edit Flat Profile**
    - Bad flat Info
    - Unsatisfying price of flat
  + **Add Flat**
    - Not reliable price
    - No Photo
    - Bad location
    - Bad flat name contains swear or badword
  + **Payment Method Addition**
    - Bad credit card number or account card number
    - Weak info
  + **Payment**
    - No balance
    - No allowement from banks
  + **Open Issue**
    - Wrong user url
  + **Close Issue**
    - No reason entered
  + **TimeSlot**
    - Date before today

# Subsystem Services

There are 7 subsystems in our system which is UserManager, FIRAuth, Search and Indexing, IssueManager, Payment, Reservation, Notification.

1.UserManager

There are 6 components in this subsystem; Flat, Rate, Host, Wishlist, Renter, SysAdmin.

In this subsystem, customer can edit our profile, add a flat to own wish list to show it later, if customer is a renter, he/she can give rate about how flat it is. If a customer is a host then he can add a flat to reserve by renter, host wants to change own flat's information at the same time. Also in this system, customer can send a mail to sysAdmin about any encountered bugs in the application or negative customer's comment.

2. FIRAuth

There are 6 components in this subsystem; login, register, passwordhasher, emailer, resetPassword, SessionManager.

In this subsystem, a visitor has to register this system then visitor can log in. Visitor registers with our tumbnail photo, name, surname, e-mail address, their own address information and then becomes a customer if all informations are corrects during registeration. In login page, customer logs in with e-mail, password. Customer’s password encrpyts by passwordHasher. SessionManager is responsible for keeping logged-in customer.

3.Payment

There are 4 components in this subsystem; Transaction, Credit Card, Bank Account, PaymentMethod.

A renter has a banking account to pay moner for reserved flat by credit card or bank account to transfer host's account. There are two payment method in this system; bank account, credit card. All procedures during this saves in transaction like who reserved flat A? , When ? etc.

4.IssueManager

There are 4 components in this subsystem; Issue, FeedbackIssue, Flat Issue, Customer Issue. As mentioned in UserManager subsystem, if there is any issue about system, flat, customer can open issue about them. This issue goes to sysAdmin.

5.ReservationManagement

There are 2 components in this subsystem; reservation, timeslot.

Reservation is makes by renter for not reserved flat if it's suitable for timeslot of flat. It must be empty.

6.Search and Indexing

There are 2 components in this subsystem; filter and search. Customer searches all flat by filtered where, from, to, how many people.

7.Notification Management

There are 4 components in this subsystem; IssueNotification, SchedularNotification, Messager, ReservationNotification.

If a renter makes reservation for a host's flat, then this notification sends to host to see that who makes reservation my flat. Messager shows this notification in system. Issue notification works any customer open issue.

# References

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Bruegge B. & Dutoit A.H. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. https://cloudplatform.googleblog.com/2015/05/Mobile-First-Development-in-the-Cloud.html
3. https://firebase.google.com/docs/database/web/structure-data
4. <https://firebase.google.com/docs/database/security/>
5. <http://angulartr.com/firebase-security-firebase-guvenligini-anlamak/>