­

Team Member

EC2 - Elaboration

Capstone Project

Residents Connect

2020CFSE007 – Suhail Haroon

2020CFSE020 – Ramesh T

2020CFSE018 - Rajeswari Subramanian

Contents

[**1. Problem Statement** 5](#_Toc81000975)

[**1.1 Contribution** 5](#_Toc81000976)

[**2. Use Cases** 5](#_Toc81000977)

[**2.1 Use Case: Resident Registration** 5](#_Toc81000978)

[**2.2 Use Case: Login Registration** 6](#_Toc81000979)

[**2.3 Use Case: Register community details** 6](#_Toc81000980)

[**2.4 Use Case: Chat Room Creation** 7](#_Toc81000981)

[**2.5 Use Case: Promote another member as admin in chat room** 7](#_Toc81000982)

[**2.6 Use Case: Add member in chat room** 8](#_Toc81000983)

[**2.7 Use Case: Member of chat room sends message in chat room** 8](#_Toc81000984)

[**2.8 Use Case: Classifieds Search** 8](#_Toc81000985)

[**2.9 Use Case: View Classifieds** 9](#_Toc81000986)

[**2.10 Use Case: Create Classifieds** 9](#_Toc81000987)

[**2.11 Use Case: Feedback to classifieds** 10](#_Toc81000988)

[**2.12 Use Case: Search facility** 10](#_Toc81000989)

[**2.13 Use Case: View Facility Booking details** 11](#_Toc81000990)

[**2.14 Use Case: Book Facility** 11](#_Toc81000991)

[**2.15 Use Case: Cancel Facility Booking** 11](#_Toc81000992)

[**2.16 Use Case: Add Announcement** 12](#_Toc81000993)

[**2.17: Use Case: Edit / View Announcement** 12](#_Toc81000994)

[**2.18 Use Case: Enroll CarPooling** 13](#_Toc81000995)

[**2.19 Use Case: Book Facility** 14](#_Toc81000996)

[**2.20 Use Case: View Slots** 14](#_Toc81000997)

[**2.21 Use Case: Add Event** 15](#_Toc81000998)

[**2.22 Use Case: View/Edit Event** 15](#_Toc81000999)

[**2.23 Use Case: Start Poll** 16](#_Toc81001000)

[**2.24 Use Case: Share / View Polls** 17](#_Toc81001001)

[**2.25 Use Case: Generate PaymentService** 17](#_Toc81001002)

[**2.26 Use Case: View PaymentService** 18](#_Toc81001003)

[**2.27 Use Case: Issue MaintenanceService** 19](#_Toc81001004)

[**2.29 Use Case: Close Maintenance** 19](#_Toc81001005)

[**2.30 Use Case: View Carpooling** 20](#_Toc81001006)

[**2.31 Use Case: Add Contacts** 20](#_Toc81001007)

[**2.32 Use Case: Edit Contacts** 21](#_Toc81001008)

[**3. Domain Model** 23](#_Toc81001009)

[**4. Logical Architecture** 32](#_Toc81001010)

[**4.1 Logical Architecture Elements** 33](#_Toc81001011)

[**4.2 Benefits of using MVC – Architecture** 34](#_Toc81001012)

[**4.3 3-tier Architecture vs. MVC Architecture** 35](#_Toc81001013)

[**5. Design Model** 35](#_Toc81001014)

[**5.1 System Design** 35](#_Toc81001015)

[**5.2 Authentication Service** 37](#_Toc81001016)

[**5.2.1 Registration** 37](#_Toc81001017)

[**5.2.2 Login** 38](#_Toc81001018)

[**5.3 Notification Service:** 38](#_Toc81001019)

[**5.4 Deployment Architecture:** 39](#_Toc81001020)

[**5.5** **CICD Pipeline:** 41](#_Toc81001021)

[**5.5.1 CICD pipeline for microservices:** 41](#_Toc81001022)

[**5.5.2 CICD pipeline for Frontend:** 41](#_Toc81001023)

[**6.** **Data Model** 42](#_Toc81001024)

[**6.1 CommunityInfo & Apartment Info Collection:** 42](#_Toc81001025)

[**6.2 UserInfo Collection** 43](#_Toc81001026)

[**6.3 ContactInfo Collection:** 44](#_Toc81001027)

[**6.4 AnnouncementInfo Collection:** 44](#_Toc81001028)

[**6.5 CarpoolingInfo Collection** 45](#_Toc81001029)

[**6.6 ChatRoomInfo Collection** 45](#_Toc81001030)

[**6.7 ClassifiedsInfo Collection** 45](#_Toc81001031)

[**6.8. EventsmgmtInfo Collection** 46](#_Toc81001032)

[**6.9 MaintenanceInfo Collection** 47](#_Toc81001033)

[**6.10. PaymentsInfo Collection** 47](#_Toc81001034)

[**6.11. PollingInfo Collection** 48](#_Toc81001035)

[**6.12. FacilityInfo Collection** 48](#_Toc81001036)

[**7. Prototyping** 48](#_Toc81001037)

# **1. Problem Statement**

Knowledge of software architectures and design as applied in the current Project will be accessed. Identification of the frontend/backend/database/cloud technologies, complete system architecture, modular decomposition of the system, block diagram, design diagrams as required to explain the system. Choice of various components of the system with justifications will be evaluated. Role of each component should be clearly elaborated.

For major technological challenges, the solution /alternatives should have been identified. Risky areas/features should have been addressed through a quick and dirty prototyping.

Note: You may not be able to implement the ideal architecture identified at this stage due to the time constrains or other resource constraints. Attempt has to be made to implement the system as close as possible to the developed architecture. Wherever this is not possible due to time or resource constraints, highlight the changes proposed for the development of the prototype.

Expected Artifact –

* Revised use cases
* Revised domain model
* Proposed Logical Architecture
* Design model
* Data model
* Prototype I (Executable)

## **1.1 Contribution**

* + 1. Use cases: Rajeswari (2020CFSE018), Suhail (2020CFSE007)
    2. Domain model: Suhail (2020CFSE007)
    3. Logical Architecture: Ramesh (2020CFSE020)
    4. Design model: Rajeswari (2020CFSE018)
    5. Data model: Rajeswari (2020CFSE018)
    6. Prototype I (Executable): Rajeswari (2020CFSE018)

# **2. Use Cases**

## **2.1 Use Case: Resident Registration**

**Goal in Context:** Resident can register to ‘ResidentConnect’ successfully.

**Scope:** Level: 1

**Preconditions:** Resident received token id for registration

**Success End Condition:** Resident enrolled to application successfully

**Failed End Condition:** Failed to enroll to application

**Primary Actor:** Resident

**Trigger:** Resident belongs to community who opted for ‘residentsconnect’ application

**Main Success Scenario:** Resident registered to application successfully

**Superordinate Use Case:**

**Subordinate Use Cases:** Login, Add Profile, Add Community

**Channel to primary actor:** Laptop, mobile

## **2.2 Use Case: Login Registration**

**Goal in Context:** Resident can login to ‘ResidentConnect’ successfully.

**Scope:** Level: 1

**Preconditions:** Resident already enrolled to application

**Success End Condition:** Resident logged in to application successfully

**Failed End Condition:** Failed to log in to application

**Primary Actor:** Resident

**Trigger:** Resident belongs to community who opted for ‘residentsconnect’ application and wants to login

**Main Success Scenario:** Resident logged in to application successfully

**Superordinate Use Case:** Registration

**Subordinate Use Cases:** Add Profile, Add Community

**Channel to primary actor:** Laptop, mobile

## **2.3 Use Case: Register community details**

**Goal in Context:** Admin can provide community details like, address, number of floors, blocks, flat details

**Scope:** Level: 1

**Preconditions:** community opted for ‘residentsconnect; application

**Success End Condition:** Admin provides community details like, address, number of floors, blocks, flat details

**Failed End Condition:** Failed to enter community details in application

**Primary Actor:** Admin

**Trigger:** Admin belongs to community who opted for ‘residentsconnect’ application and wants to provide community details on first time usage of the application for that community

**Main Success Scenario:** Admin provides community details like, address, number of floors, blocks, flat details

**Superordinate Use Case:** Login, Registration

**Subordinate Use Cases:** Add Profile, Add Community

**Channel to primary actor:** Laptop, mobile

## **2.4 Use Case: Chat Room Creation**

**Goal in Context:** Resident can create chat room

**Scope:** Level: 1

**Preconditions:** resident enrolled to application

**Success End Condition:** Resident created chat room

**Failed End Condition:** Failed to create chat room

**Primary Actor:** Resident

**Trigger:** Resident belongs to community who opted for ‘residentsconnect’ application and wants to create chat room to talk with other members in that community.

**Main Success Scenario:** Resident created chat room

**Superordinate Use Case:** Login, Registration

**Subordinate Use Cases:** **:** Member of chat room sends message in chat room, promote other member as admin in chat

**Channel to primary actor:** Laptop, mobile

## **2.5 Use Case: Promote another member as admin in chat room**

**Goal in Context:** Resident who is admin of the chat room can promote another member in chat group as admin

**Scope:** Level: 1

**Preconditions:** resident enrolled to application, and admin of the chat room

**Success End Condition:** Resident can promote another member as admin

**Failed End Condition:** Failed to promote another member as admin

**Primary Actor:** Resident & Admin of the chat room

**Trigger:** Received the request from another member in chat group to convert him to admin

**Main Success Scenario:** Resident promotes another member as admin

**Superordinate Use Case:** create chat room

**Subordinate Use Cases:** **:** Member of chat room sends message in chat room

**Channel to primary actor:** Laptop, mobile

## **2.6 Use Case: Add member in chat room**

**Goal in Context:** Admin/Member of chat room can add another resident in chat room

**Scope:** Level: 1

**Preconditions:** resident enrolled to application, and raised request add him to as member in a chat room

**Success End Condition:** Admin/Member of the chat room can add another resident in the chat room

**Failed End Condition:** Failed add another resident in the chat room

**Primary Actor:** Resident & member of chat room

**Trigger:** Resident wants to communicate with other member in the chat room

**Main Success Scenario:** Admin/Member of the chat room adds another resident in the chat room

**Superordinate Use Case:**  Chat Room Creation

**Subordinate Use Cases:** Promote another member as admin in chat room

**Channel to primary actor:** Laptop, mobile

## **2.7 Use Case: Member of chat room sends message in chat room**

**Goal in Context:** Member in chat room can able to send messages in chat room (it may be text, file, video, image, etc..)

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application, and member of the chat room

**Success End Condition:** Resident can send the message in chat room

**Failed End Condition:** Failed to send the message in chat room

**Primary Actor:** Resident

**Trigger:** resident wants to communicate with other member in the chat room

**Main Success Scenario:** Resident sends message in chat room

**Superordinate Use Case:**  Add member in chat room

**Subordinate Use Cases:** **:** Promote another member as admin in chat room

**Channel to primary actor:** Laptop, mobile

## **2.8 Use Case: Classifieds Search**

**Goal in Context:** Resident can search classifieds based on category, subcategory & Ratings

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application

**Success End Condition:** Application can return list of classifieds based on search criteria.

**Failed End Condition:** Classifieds are not returned based on search criteria.

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants to know about some service offered by near by area

**Main Success Scenario:** Application returns list of classifieds based on search criteria.

**Superordinate Use Case:** Registration, Login

**Subordinate Use Cases:**  View Classifieds, Create Classifieds, Feedback to classifieds.

**Channel to primary actor:** Laptop, mobile

## **2.9 Use Case: View Classifieds**

**Goal in Context:** Resident can view classified

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application

**Success End Condition:** Resident can view classified by clicking details link next to classified in ‘Classifieds Search’ page

**Failed End Condition:** Classified details are not shown

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants to know about some service offered by nearby area

**Main Success Scenario:** Resident view classified by clicking details link next to classified in ‘Classifieds Search’ page

**Superordinate Use Case:** Registration, Login, Classifieds Search

**Subordinate Use Cases:**  Create Classifieds, Feedback to classifieds.

**Channel to primary actor:** Laptop, mobile

## **2.10 Use Case: Create Classifieds**

**Goal in Context:** Resident can create classifieds

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application and wants to post some advertisement for service

**Success End Condition:** Resident can create classified

**Failed End Condition:** Failed to create classified.

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants to create classified

**Main Success Scenario:** Resident creates classified

**Superordinate Use Case:** Registration, Login,

**Subordinate Use Cases:**  Feedback to classifieds, Classifieds Search

**Channel to primary actor:** Laptop, mobile

## **2.11 Use Case: Feedback to classifieds**

**Goal in Context:** Resident can provide feedback to classifieds

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application and wants to provide feedback to service offered by classified’s company

**Success End Condition:** Resident can provide feedback to classified & rate it

**Failed End Condition:** Failed to provide feedback

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants to provide feedback

**Main Success Scenario:** Resident provides feedback to classified & rate it

**Superordinate Use Case:** Registration, Login, View, Classifieds Search, View Classified

**Subordinate Use Cases:**  Create Classified

**Channel to primary actor:** Laptop, mobile

## **2.12 Use Case: Search facility**

**Goal in Context:** Resident can search facility details like ‘swimming pool’, ‘Party Hall’

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application and wants book facility for his use

**Success End Condition:** Resident can view facility details based on his search criteria

**Failed End Condition:** Failed to view facility details

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants book facility for his use

**Main Success Scenario:** Resident books facility for his use

**Superordinate Use Case:** Registration, Login

**Subordinate Use Cases:**  View Facility Booking details, Book Facility, Cancel Facility Booking

**Channel to primary actor:** Laptop, mobile

## **2.13 Use Case: View Facility Booking details**

**Goal in Context:** Resident can view booking details of the facility

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application and wants book facility for his use

**Success End Condition:** Resident can view booking details of the facility

**Failed End Condition:** Failed to view facility booking details

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants book facility for his use

**Main Success Scenario:** booking details of the facility are shown in application

**Superordinate Use Case:** Registration, Login, Search facility

**Subordinate Use Cases:**  Book Facility, Cancel Facility Booking

**Channel to primary actor:** Laptop, mobile

## **2.14 Use Case: Book Facility**

**Goal in Context:** Resident can book facility on particular day and time slot

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application and wants book facility for his use

**Success End Condition:** Resident can book facility on particular day and time slot

**Failed End Condition:** Resident failed to book facility on particular day and time slot

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants book facility for his use

**Main Success Scenario:** Resident book facility on particular day and time slot

**Superordinate Use Case:** Registration, Login, Search facility, View Facility Booking details

**Subordinate Use Cases:**  Cancel Facility Booking

**Channel to primary actor:** Laptop, mobile

## **2.15 Use Case: Cancel Facility Booking**

**Goal in Context:** Resident can cancel facility booking

**Scope:** Level: 1

**Preconditions:** Resident enrolled to application and wants cancel facility booking

**Success End Condition:** Resident can cancel facility booking

**Failed End Condition:** Resident failed to cancel booking

**Primary Actor:** Enrolled Resident

**Trigger:** Resident wants cancel facility booking

**Main Success Scenario:**  Resident cancel facility booking

**Superordinate Use Case:** Registration, Login, Search facility, View Facility Booking details, Book Facility

**Subordinate Use Cases:**

**Channel to primary actor:** Laptop, mobile

## **2.16 Use Case: Add Announcement**

Goal in Context: Announcement to the public/Invites

Scope: Announcement Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Create Announcement to the public / Invitees

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: Admin

Trigger: Accessing Add Announcements functionality from service

Announcement will be hosted and added for all invited people to see

step 1 Creation of Announcement

step 2 Display of Announcement to target Audience: All nearby if public announcement, Else invited people if invite only event

Superordinate Use Case:

Create Announcement, View Announcement, Edit Announcement, Cancel Announcement

Channel to primary actor: Via Announcement micro service

## **2.17: Use Case: Edit / View Announcement**

Goal in Context: Edit Announcement or View - Published to the public/Invites

Scope: Announcement Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Edit /View Announcement to the public / Invitees

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: User

Trigger: Accessing Deleting Announcements functionality from service

**MAIN SUCCESS SCENARIO**

Announcement will be hosted and added for all invited people to see

step 1 Edit / View of Announcement

step 2 Display of Announcement to target Audience: All nearby if public event, Else invited people if invite only event

Superordinate Use Case:

Subordinate Use Cases: View Announcement, Edit Announcement, Cancel Announcement

Channel to primary actor: Via Announcement micro service

## **2.18 Use Case: Enroll CarPooling**

**CHARACTERISTIC INFORMATION**

Goal in Context: carpooling to the public/Invites

Scope: Carpooling Service Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Create CarpoolingService to the public / Invitees

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Enroll Carpooling functionality from service

**MAIN SUCCESS SCENARIO**

Classifieds will be hosted and added for all invited people to see

step 1 Creation of Carpooling

step 2 Display of Carpooling to target Audience: All nearby if public, Else invited people if invite only event

Superordinate Use Case:

Subordinate Use Cases: Create Carpooling, View Carpooling

Channel to primary actor: Via Carpooling micro service

## **2.19 Use Case: Book Facility**

**CHARACTERISTIC INFORMATION**

Goal in Context: FacilityBookingService to the public/Invites

Scope: FacilityBookingService Service Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Create FacilityBookingService to the public / Invitees

Failed End Condition: Error returned to user from service

Primary Actor: Admin/User

Trigger: Book FacilityBooking functionality from service

**MAIN SUCCESS SCENARIO**

Classifieds will be hosted and added for all invited people to see

step 1 Creation of Facility Booking

step 2 Display Facility Booked for the respective facility and slot.

Superordinate Use Case:

Subordinate Use Cases: Create FacilityBooking , View FacilityBooking

Channel to primary actor: Via FacilityBooking micro service

## **2.20 Use Case: View Slots**

**CHARACTERISTIC INFORMATION**

Goal in Context: View – available slots for respective facility

Scope: FacilityBookingService Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: View Slots available for booking facility

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Accessing Deleting FacilityBookingService functionality from service

**MAIN SUCCESS SCENARIO**

Classifieds will be hosted and added for all invited people to see

step 1 View of available slots

step 2 Display of Facility Booking to residents of the community with the facilities

Superordinate Use Case:

Subordinate Use Cases: View FacilityBookingService

Channel to primary actor: Via Facility Booking micro service

## **2.21 Use Case: Add Event**

**CHARACTERISTIC INFORMATION**

Goal in Context: Adding/Hosting an Event to the the public or with Invites

Scope: Event Management Micro service

Level: 2

Preconditions: Accessing add event

Success End Condition: Created Event to the public

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Accessing Add event functionality from service

**MAIN SUCCESS SCENARIO**

Event will be hosted and added to Nearby Events for all invited people to see

step 1 Creation of Event

step 2 Display of Event to target Audience: All nearby if public event, Else invited people if invite only event

Superordinate Use Case:

Subordinate Use Cases: Enroll Event, View Events, Edit Event, Resitered Event

Channel to primary actor: Via Event management micro service

## **2.22 Use Case: View/Edit Event**

**CHARACTERISTIC INFORMATION**

Goal in Context: Manage/View existing added Event by user to the the public or with invites

Scope: Event Management Micro service

Level: 2

Preconditions: Accessing Manage event

Success End Condition: View/Update Event to the public

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Accessing Edit event functionality from service

**MAIN SUCCESS SCENARIO**

Event will be hosted and added to Nearby Events for all invited people to see

step 1 Updating of Event

step 2 Display of Event to target Audience: All nearby if public event, Else invited people if invite only event

Superordinate Use Case:

Subordinate Use Cases: Enroll Event, Add Event, Registered Event

Channel to primary actor: Via Event management micro service

## **2.23 Use Case: Start Poll**

**CHARACTERISTIC INFORMATION**

Goal in Context: Start poll within community

Scope: Polling Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Create Polls to the community

Failed End Condition: Error returned to user from service

Primary Actor: Admin/User

Trigger: Add Polls functionality from service

**MAIN SUCCESS SCENARIO**

Polls will be added for all residents in the community to see

step 1 Creation of Polls

step 2 Display of Polls to target Audience:

Superordinate Use Case:

Subordinate Use Cases: Create Polls, View Polls

Channel to primary actor: Via Polling micro service

## **2.24 Use Case: Share / View Polls**

**CHARACTERISTIC INFORMATION**

Goal in Context: Share/ View created polls

Scope: Polling Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Share /View Polls within the community

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: User

Trigger: Accessing View Polls functionality from service

**MAIN SUCCESS SCENARIO**

viewing created polls for all residents within the community to see

step 1 Edit / View of Polls

step 2 Display of Polls to the residents in the community

Superordinate Use Case:

Subordinate Use Cases: Add Polls

Channel to primary actor: Via Polling micro service

## **2.25 Use Case: Generate PaymentService**

**CHARACTERISTIC INFORMATION**

Goal in Context: PaymentService to the public/Invites

Scope: PaymentService Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Create PaymentService to the public / Invitees

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: Admin/User

Trigger: Accessing Generate PaymentService from service

**MAIN SUCCESS SCENARIO**

Announcement will be hosted and added for all invited people to see

step 1 Creation of PaymentService

step 2 Display of PaymentService to target Audience: All nearby if PaymentService, Else invited people if invite only event

Superordinate Use Case:

Subordinate Use Cases: Generate PaymentService, View PaymentService

Channel to primary actor: Via PaymentService micro service

## **2.26 Use Case: View PaymentService**

**CHARACTERISTIC INFORMATION**

Goal in Context: View PaymentService - Published to the public/Invites

Scope: PaymentService Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: View PaymentService to the public / Invitees

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: Admin/User

Trigger: Accessing Deleting PaymentService functionality from service

**MAIN SUCCESS SCENARIO**

Announcement will be hosted and added for all invited people to see

step 1 View of PaymentService

step 2 Display of PaymentService to target Audience: All nearby if PaymentService , Else invited people if invite only event

Superordinate Use Case:

Subordinate Use Cases: View PaymentService

Channel to primary actor: Via PaymentService micro service

## **2.27 Use Case: Issue MaintenanceService**

**CHARACTERISTIC INFORMATION**

Goal in Context: MaintenanceService to the public/Invites

Scope: MaintenanceService Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Create MaintenanceService to the public / Invitees

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: User

Trigger: Accessing request Maintenance from service

**MAIN SUCCESS SCENARIO**

Announcement will be hosted and added for all invited people to see

step 1 Added Maintenance request

step 2 Display of Maintenance for the respective person conveying request and admin

Superordinate Use Case:

Subordinate Use Cases: Issue Maintenance request, Close Maintenance request

Channel to primary actor: Via Maintenance micro service

## **2.29 Use Case: Close Maintenance**

**CHARACTERISTIC INFORMATION**

Goal in Context: Close Maintenance requests

Scope: Maintenance Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Close Maintenance to the public / Invitees

Failed End Condition: Error returned to user from service, Expired At

Primary Actor: User

Trigger: Accessing Deleting Maintenance functionality from service

**MAIN SUCCESS SCENARIO**

Announcement will be hosted and added for all invited people to see

step 1 Close of Maintenance request

step 2 Closed maintenance request posted by user

Superordinate Use Case:

Subordinate Use Cases: View MaintenanceService

Channel to primary actor: Via MaintenanceService micro service

## **2.30 Use Case: View Carpooling**

**CHARACTERISTIC INFORMATION**

Goal in Context: View - Carpooling Published to the public/Invites

Scope: Carpooling Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: View Carpooling to the public / Invitees

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Accessing Carpooling functionality from service

**MAIN SUCCESS SCENARIO**

Classifieds will be hosted and added for all invited people to see

step 1 View of Carpooling

step 2 Display of Carpooling to target Audience: All nearby if public, Else invited people if invite only event

**Superordinate Use Case:**

Subordinate Use Cases: View Carpooling

Channel to primary actor: Via Carpooling micro service

## **2.31 Use Case: Add Contacts**

**CHARACTERISTIC INFORMATION**

Goal in Context: Contacts to the public/Invites

Scope: Contacts Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Added Contacts for public access within the community

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Accessing Add Contacts from services

**MAIN SUCCESS SCENARIO**

Contact will be updates for all invited people to see

step 1 Creation of Contacts

step 2 Display of Contacts to residents of the community

Superordinate Use Case:

Subordinate Use Cases: Add Contact

Channel to primary actor: Via Contacts micro service

## **2.32 Use Case: Edit Contacts**

**CHARACTERISTIC INFORMATION**

Goal in Context: Edit Contacts

Scope: Contacts Micro service

Level: 2

Preconditions: Existing User or Preferred User

Success End Condition: Updated the Contacts for the residents of the community

Failed End Condition: Error returned to user from service

Primary Actor: User

Trigger: Accessing Deleting Contacts functionality from service

**MAIN SUCCESS SCENARIO**

Announcement will be hosted and added for all invited people to see

step 1 Edit ContactsService

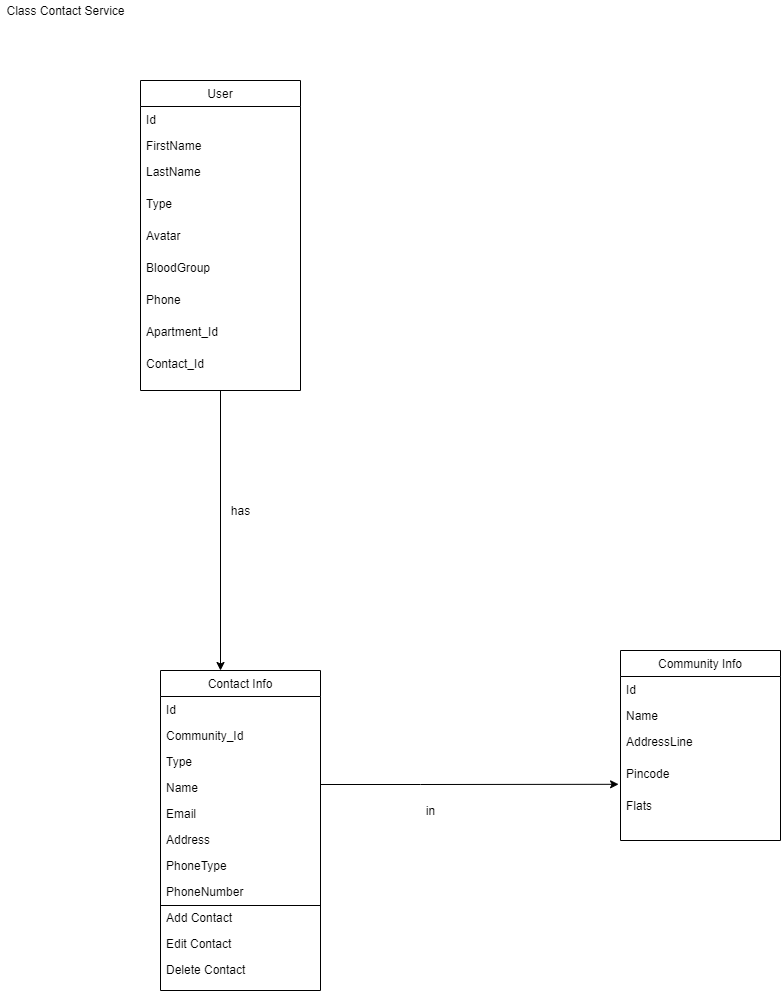
step 2 Display of Contacts for the community residents

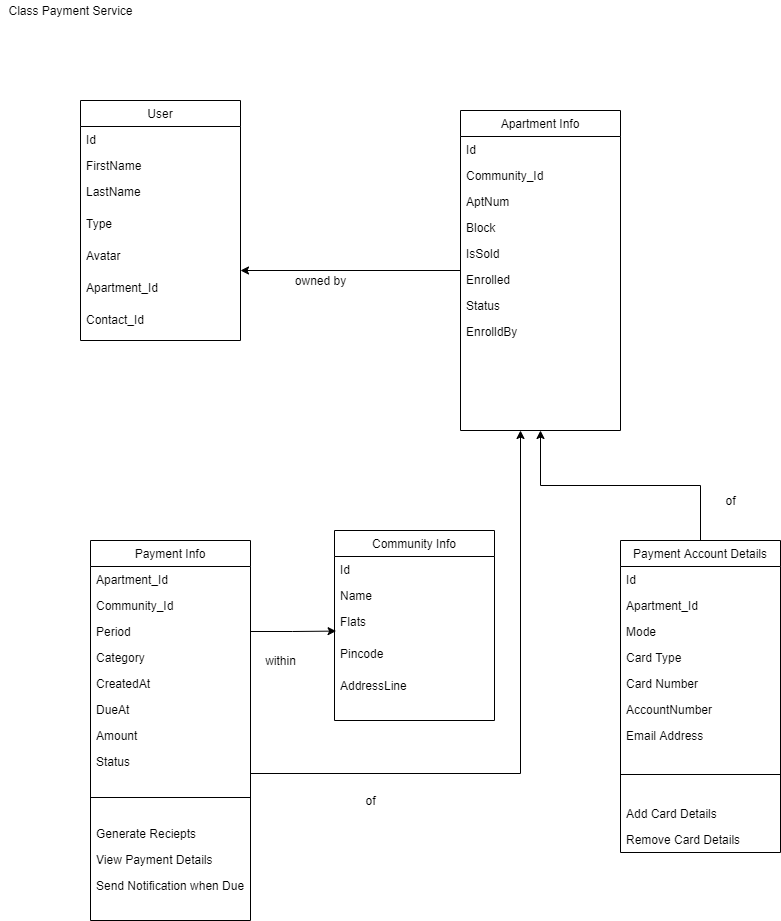
Superordinate Use Case:

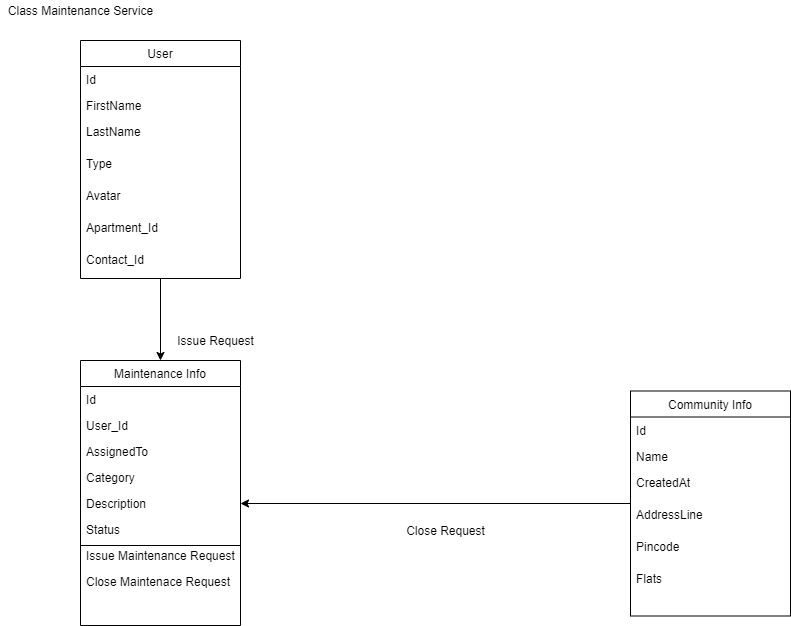
Subordinate Use Cases: Add ContactsService

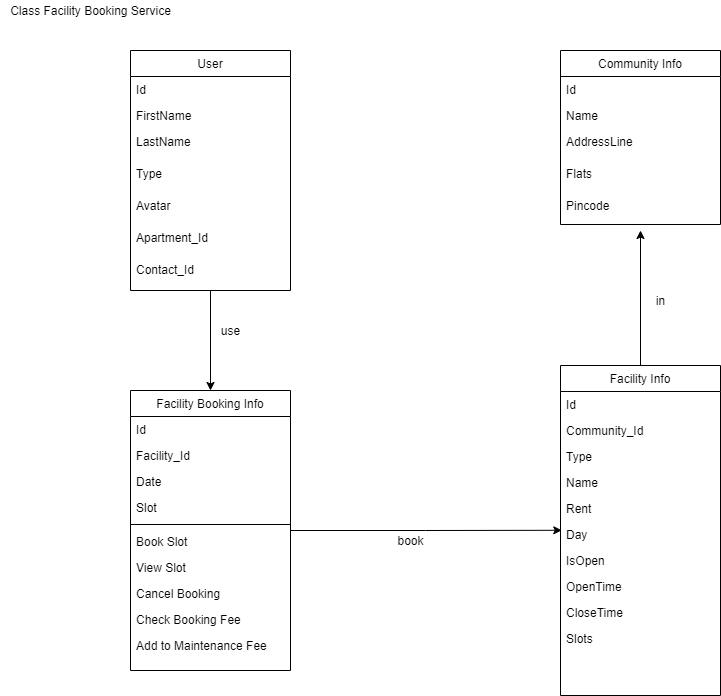
Channel to primary actor: Via Contacts micro service

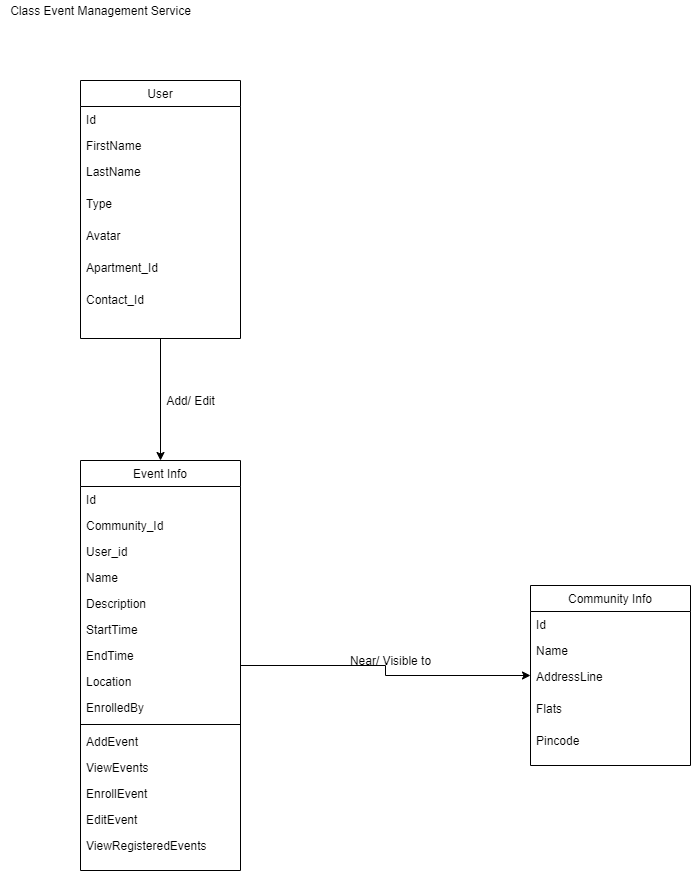
# **3. Domain Model**

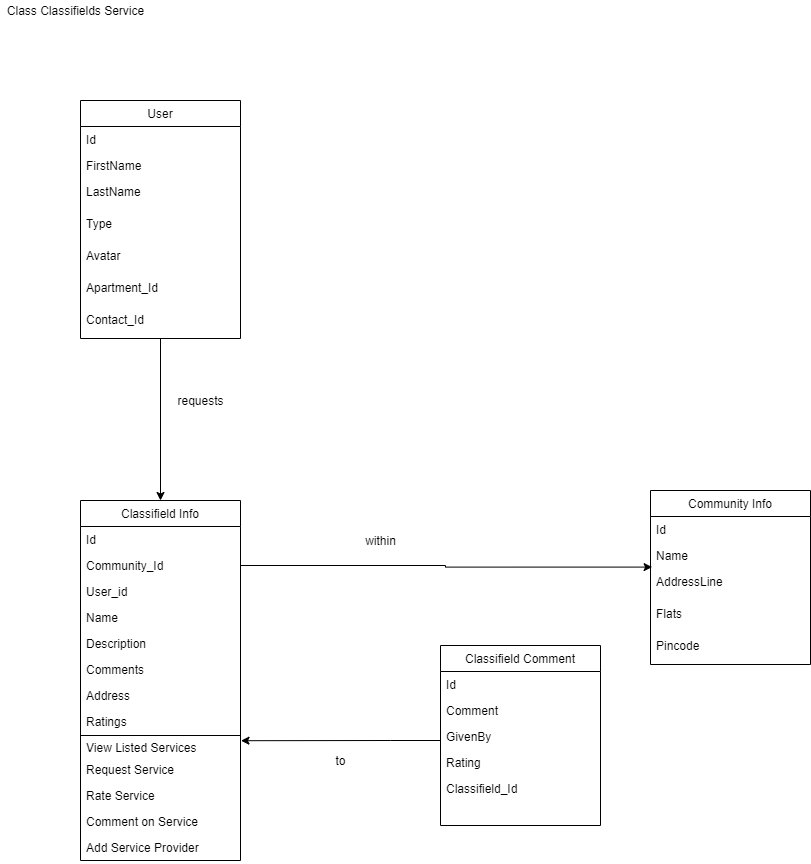


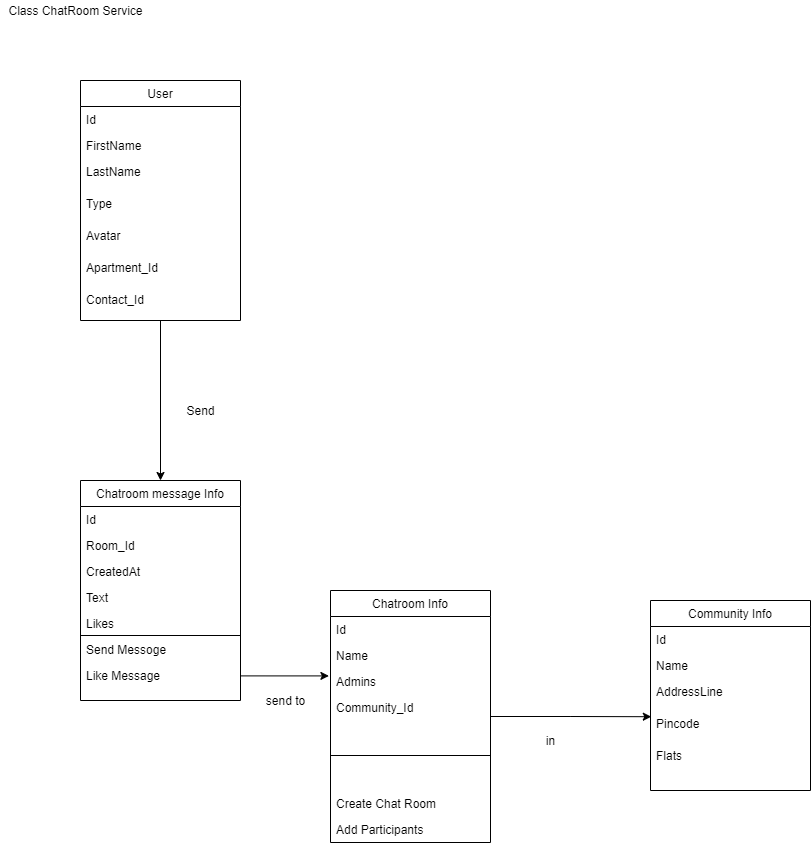


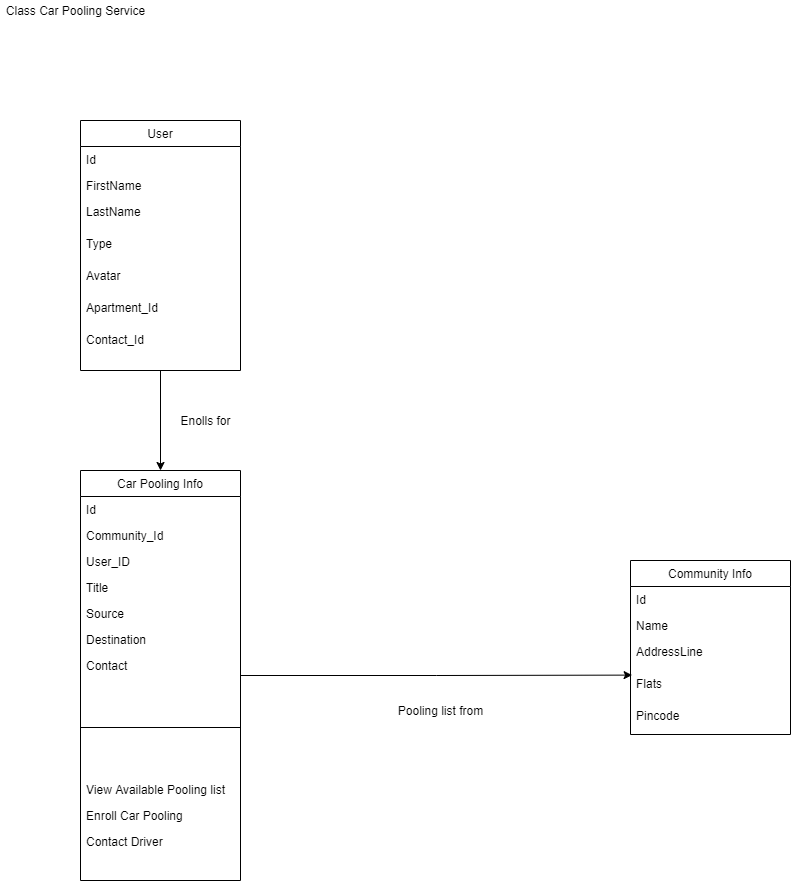


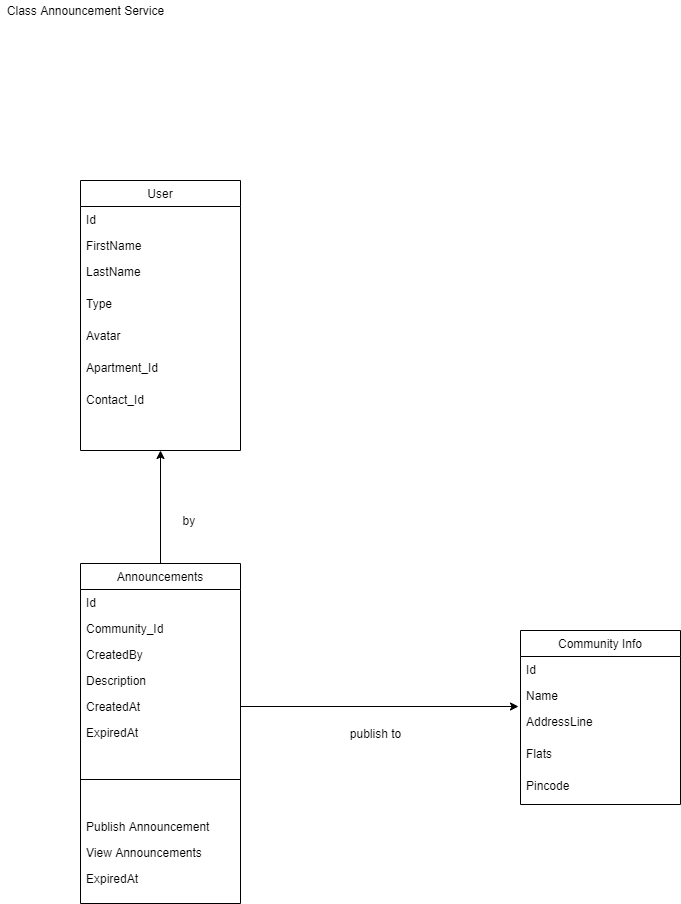


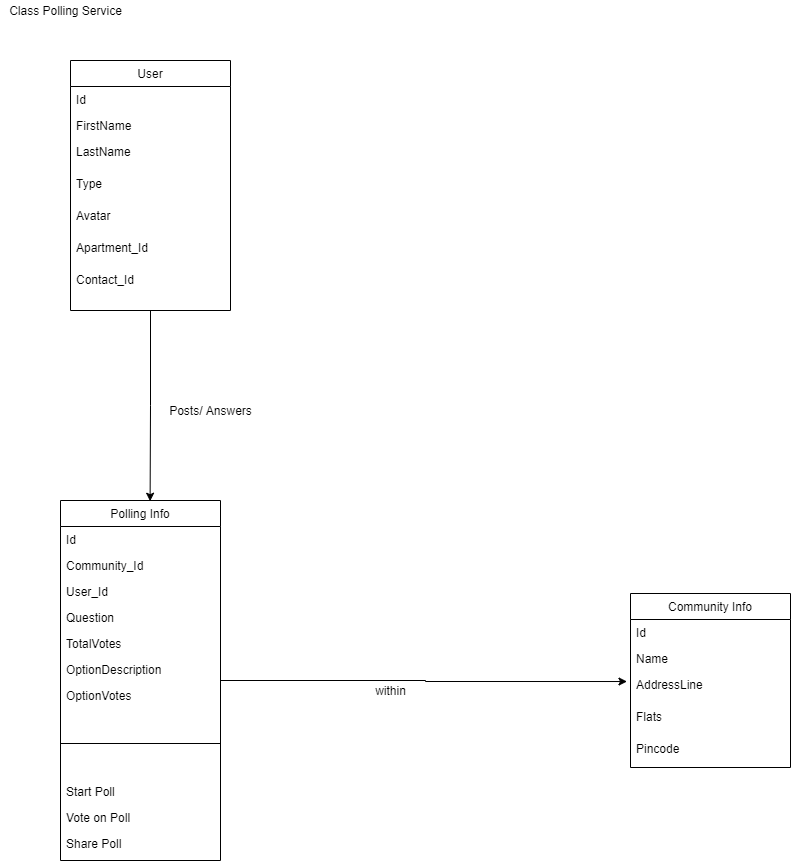




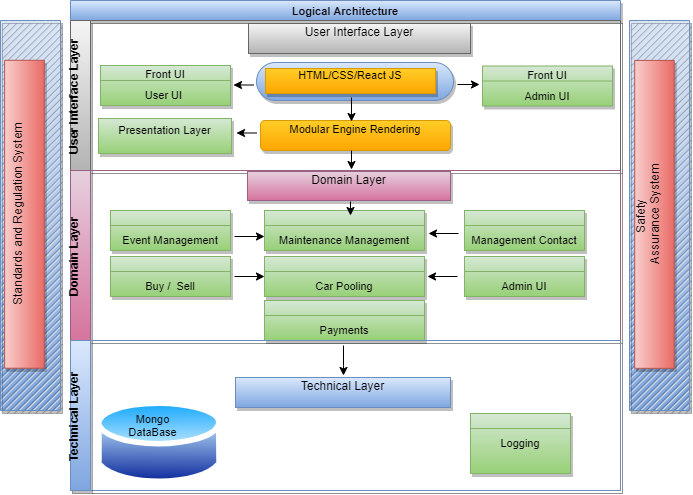








# **4. Logical Architecture**



Large scale organization of software classes into packages, subsystems and layers

• Logical grouping and no Deployment Considered

• Layered architecture

o Coarse grouping of classes, packages, subsystems that has a cohesive responsibility

o Higher layers consumes services of lower layers

• Usual layers

o User Interface

o Application Logic and Domain

o Technical Services

## **4.1 Logical Architecture Elements**

• UI  
 o view   
 o Common mobile or web interfaces

• Application

o Workflow, app controller   
o Handles requests from presentation layers   
o REST endpoint handler

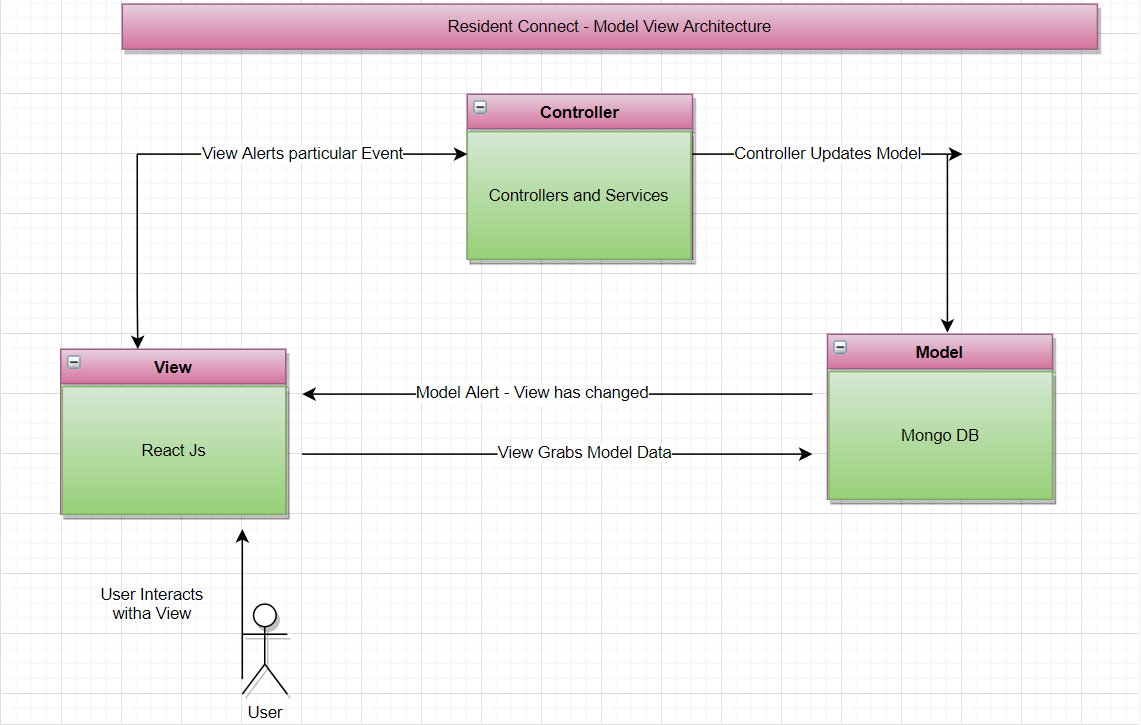
• Domain

o Business logic   
o Takes care of implementing business rules as response to app layer requests

• Business Infrastructure   
 o Very general low level business services like converters

• Technical Services   
 o High level technical services and frameworks like logger, DB adapter

• Foundations   
 o Low level technical services, frameworks, databases

****

## **4.2 Benefits of using MVC – Architecture**



* Easy code maintenance which is easy to extend and grow
* MVC Model component can be tested separately from the user
* Easier support for new types of clients
* Development of the various components can be performed parallelly.
* It helps you to avoid complexity by dividing an application into the three units. Model, view, and controller
* It only uses a Front Controller pattern which process web application requests through a single controller.
* Offers the best support for [test-driven development](https://www.guru99.com/test-driven-development.html)
* It works well for Web apps which are supported by large teams of web designers and developers.
* Provides clean separation of concerns (SoC).
* Search Engine Optimization (SEO) Friendly.
* All classes and objects are independent of each other so that you can test them separately.
* MVC design pattern allows logical grouping of related actions on a controller together.
* The MVC is an architectural pattern that separates an application into
  + 1) Model, 2) View and 3) Controller
* Model: It includes all the data and its related logic



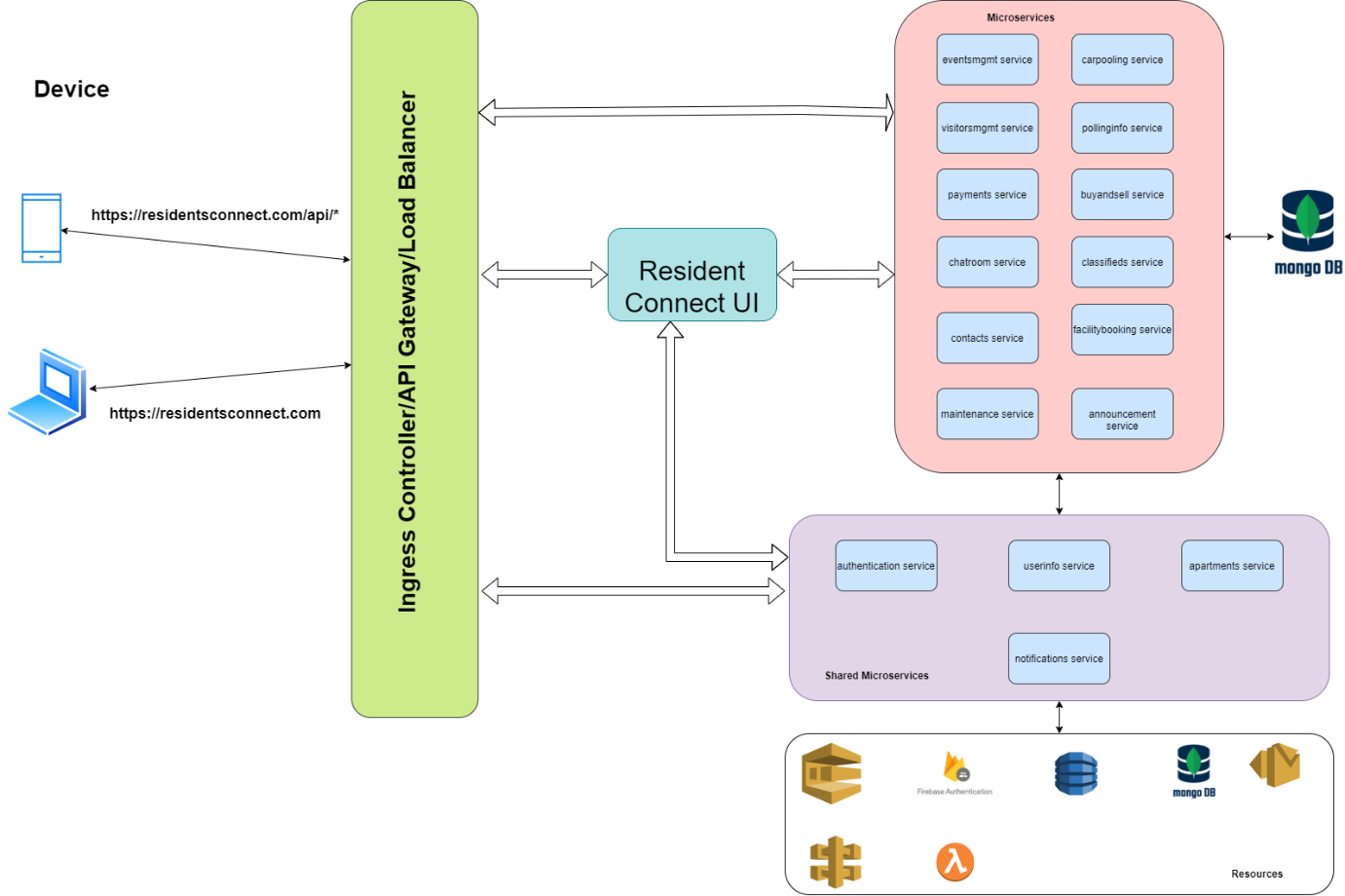
* View: Present data to the user or handles user interaction
* Controller: An interface between Model and View components

## **4.3 3-tier Architecture vs. MVC Architecture**

|  |  |  |
| --- | --- | --- |
| Parameter | 3-Tier Architecture | MVC Architecture |
| Communication | This type of architecture pattern never communicates directly with the data layer. | All layers communicate directly using triangle topology. |
| Usage | 3-tier: widely used in web applications where the client, data tiers, and middleware a run on physically separate platforms. | Generally used on applications that run on a single graphical workstation. |

# **5. Design Model**

## **5.1 System Design**



**Microservice Architecture:**

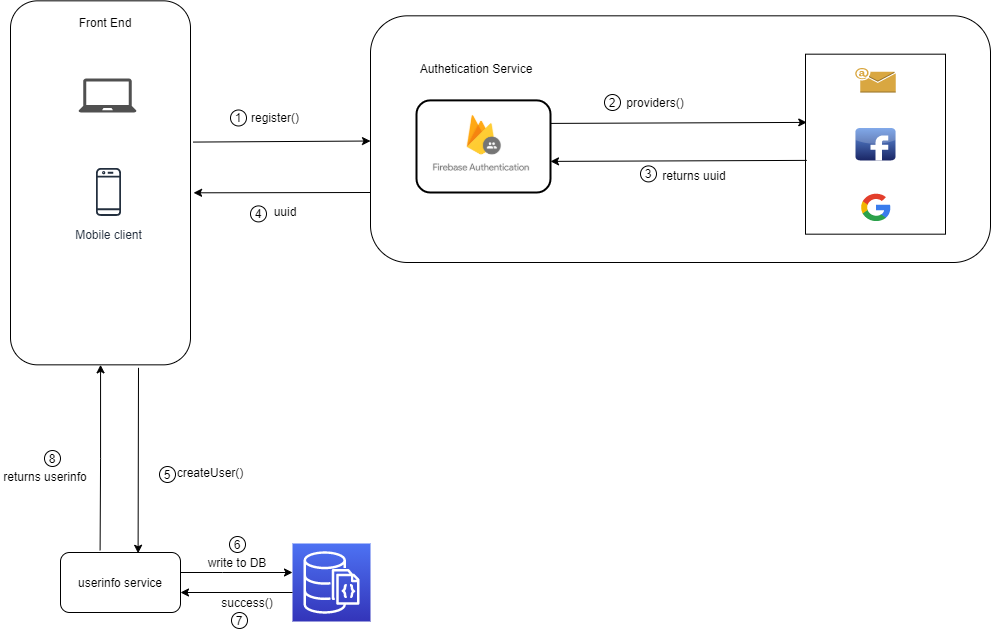
* Microservice -based architecture is considered for this project. Each and every functionality would be implemented as separate microservices. They may or may nor share the same databases though many of them using mango db.
* Event based architecture is not considered, as microservices in this application are mostly independent of one another.
* Notification & Chat room services are planned to be implemented using serverless with the help of AWS and firebase accordingly.

**Application load balancer vs API gateway:**

* Application load balancer uses to manage the network traffic based path routing. Based api path, it routes to corresponding target where micro service is running.
* API gateway : Notification service creates endpoint to send the notification which will be used by other services.

## **5.2 Authentication Service**

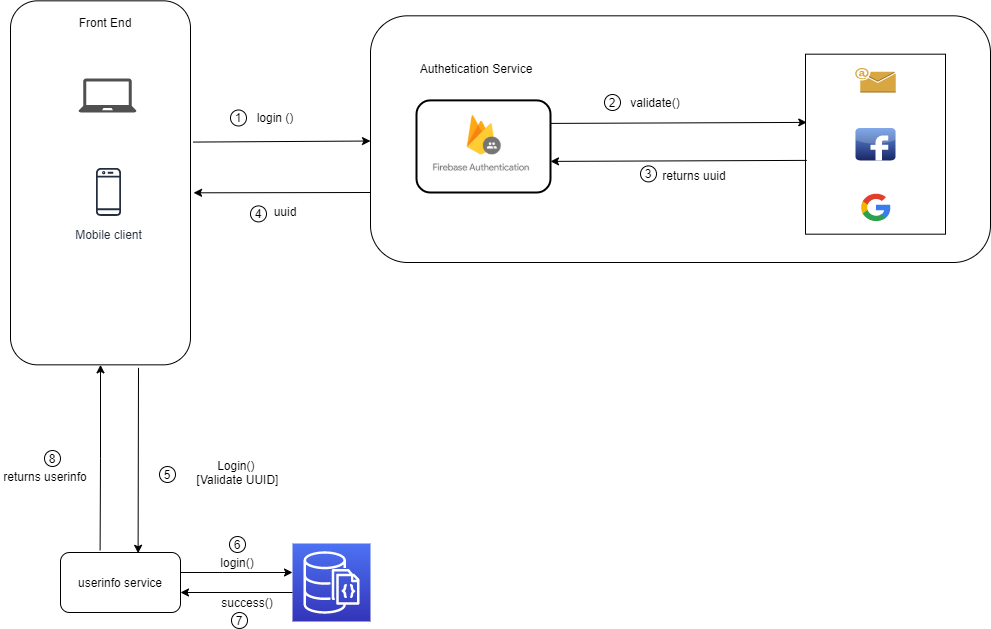
## **5.2.1 Registration**



Application invokes register method with type of registration (gmail, facebook, email/password), he prefers. Based on the request received from client, firebase authentication service registers users and return user details (along with unique UUID). Once uuid received from firebase, createuser would be made to userinfo service to store user details like email address, uuid, etc..

Other details like name, phone number, blood group while resident/admin setup profile.

## **5.2.2 Login**



During login service call, firebase returns uuid, if user is registered already. This uuid would be compared with uuid present in userinfo collection with the help of userinfo service for vertification.

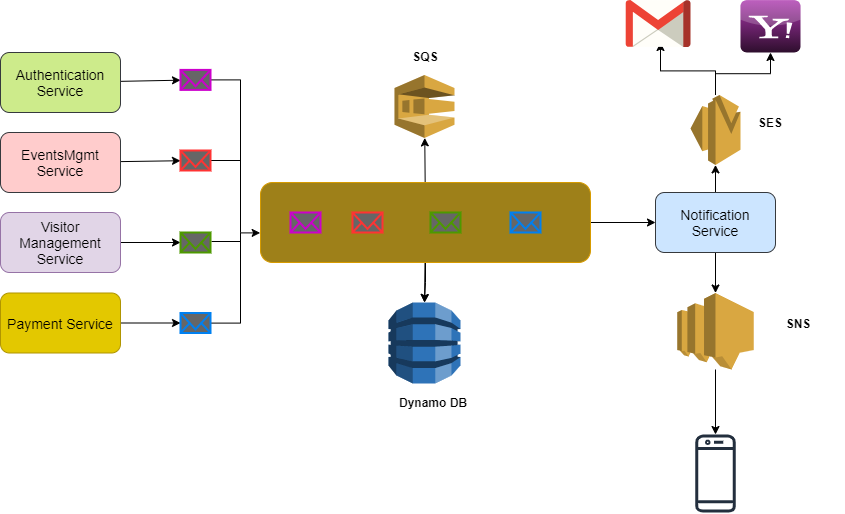
## **5.3 Notification Service:**

Email notification service is created using AWS-serverless method. SQS (Simple Queue Service), SES (Simple Email Service), Dynamo DB is used.

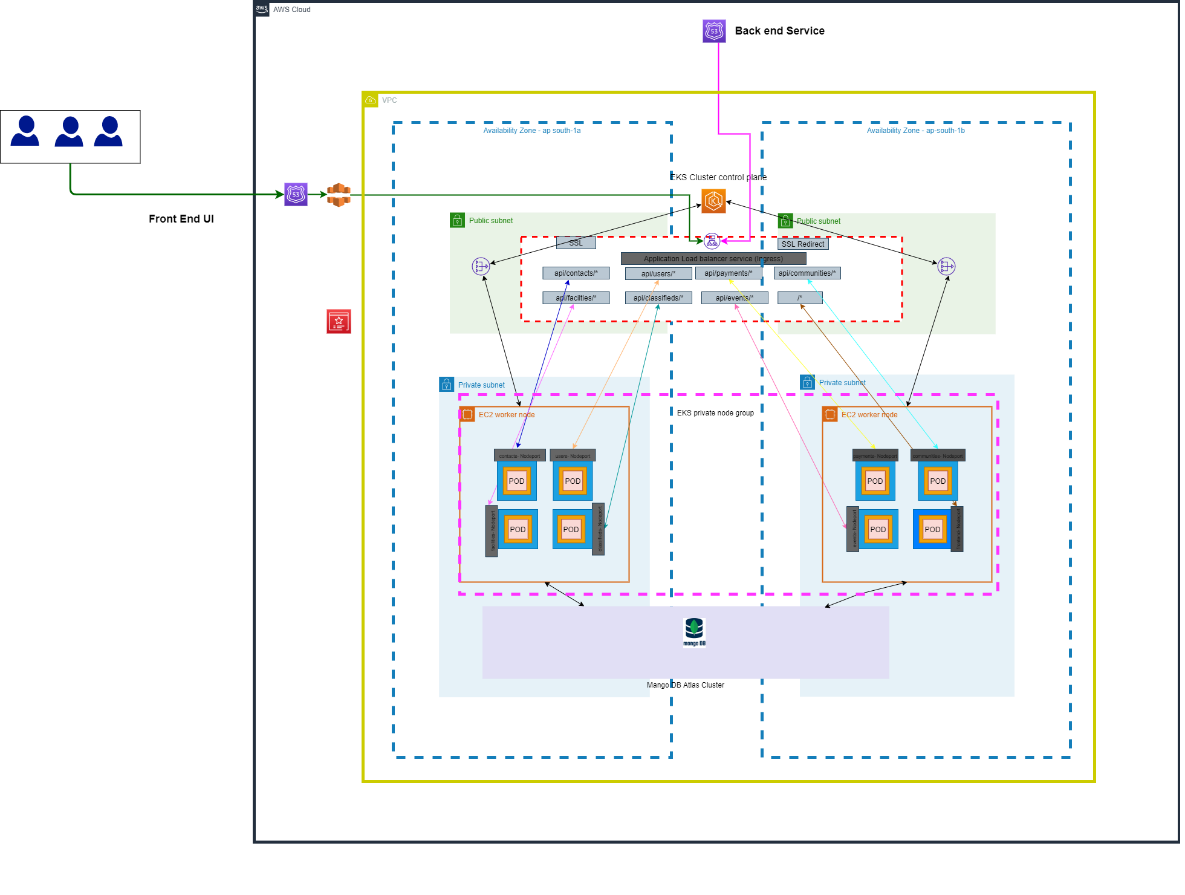
**SQS (Simple Queue Service):** Generated email message is sent to SQS queue and details are recorded in Dynamo DB. sendMail – POST API is called for this purpose. This endpoint is created by AWS gateway.

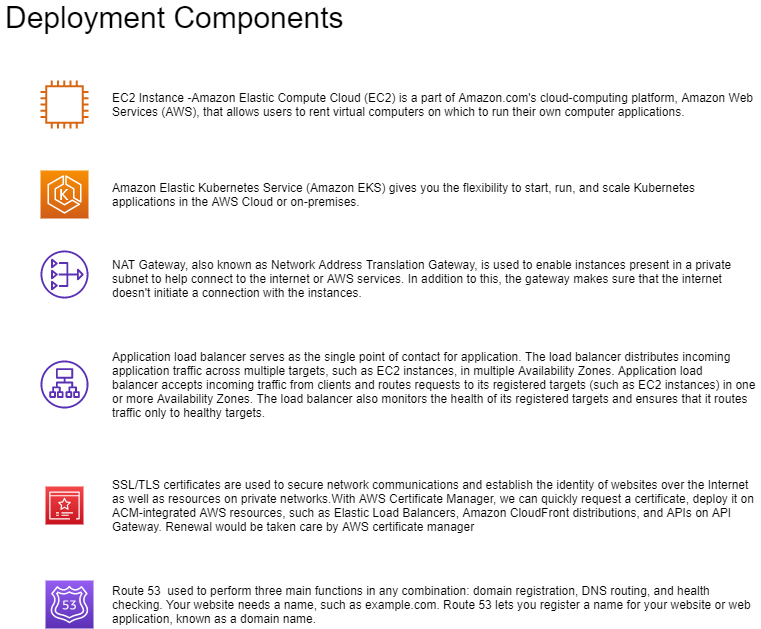
**SES (Simple Email Service) :** Notification service listens SQS using ‘receiveMail’ handler and generate email message. This generated email message sent to corresponding receipts. Any reminder mail which needs to be send, this service will take care of it.

**SNS (Simple Notification Service):** SNS service is used for sending message to resident’s mobile numbers. Using Amazon SNS topics, publisher systems can fanout messages to a large number of subscriber systems including Amazon SQS queues, AWS Lambda functions and HTTPS endpoints



## **5.4 Deployment Architecture:**





Below AWS infrastructure used for deployment,

* AWS Resources
* VPC
* Internet Gateway (IGW)
* Security Groups, Route Tables and Route Table Associations
* IAM roles, Instance profiles and Policies
* An EKS Cluster
* Autoscaling group and Launch Configuration
* EKS Worker Nodes
* ALB Ingress
* The ConfigMap required to register Nodes with EKS
* KUBECONFIG file to authenticate kubectl using the heptio authenticator aws
* binary
* Elastic Beanstalk
* MongoDB in Atlas
* Jenkins in EC2 instance
* IAM user for accessing the resources
* CloudFront

## **CICD Pipeline:**

### **5.5.1 CICD pipeline for microservices:**

Microservices are deployed to EKS using AWS codepipeline



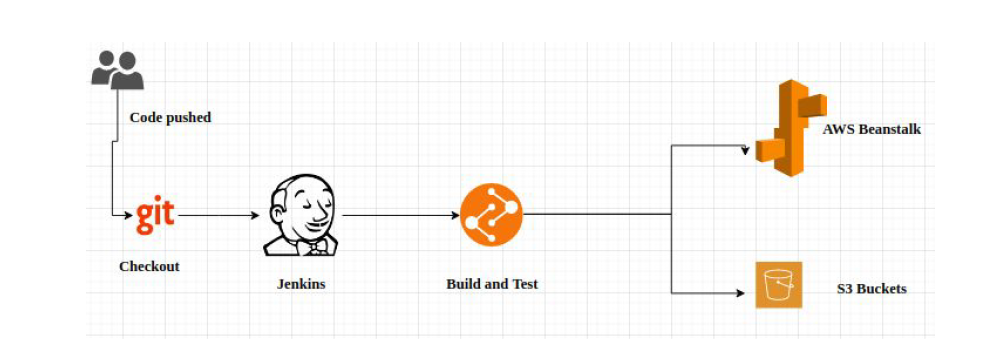
Steps involved,



* 1. User/developer pushed the code in git.
  2. “AWS Code Build “ tests microservice, It runs automation testcases
  3. When testcases passes, Docker image is created and pushed to ECR.
  4. EKS updates the deployment Pods (groups of one or more Docker containers with shared storage/network) using a rolling update strategy by picking the images from Elastic Container Registry automatically.

### **5.5.2 CICD pipeline for Frontend:**

Below jenkin pipeline used for deployment of ‘React JS front end’ code in AWS EC2 instance.



Steps involved,



* 1. User/developer pushed the code in git.
  2. Jenkin performs build& test,
     + 2.1 It runs automation testcases
     + 2.2 When testcases passes, application would be built
  3. Build would be pushed to S3 and build is taken by beanstack from S3
  4. Elastic Beanstalk will deploy the application and manage the application so that there is a very less down time. Load balancer attached with Beanstalk provides autoscaling and make sure the application is running properly and if there is high traffic load then it will spin up a new instance to balance the traffic

# **Data Model**

Non-SQL databases are used for this project , as this application involves images, videos on various phases.

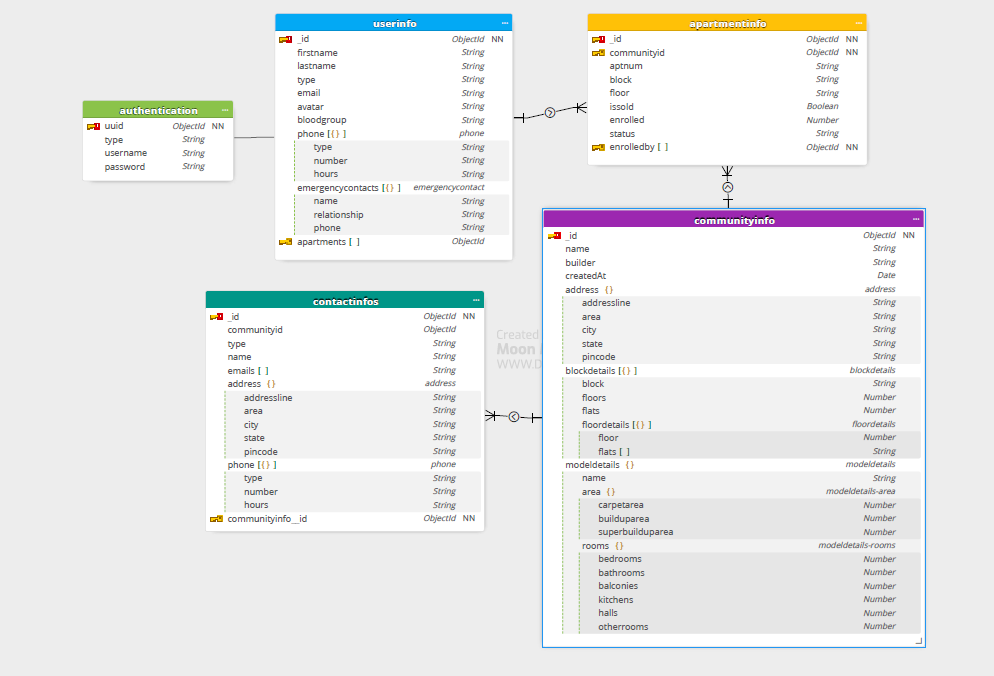
Below Non-SQL databases are used in this project

* 1. DynamoDB – Used by notification service, to store notification messages
  2. Firebase live database – This database will be used for chatroom service.
  3. Atlas MangoDB: All other services use mango DB for storing their data.

Please do refer below schema diagrams,

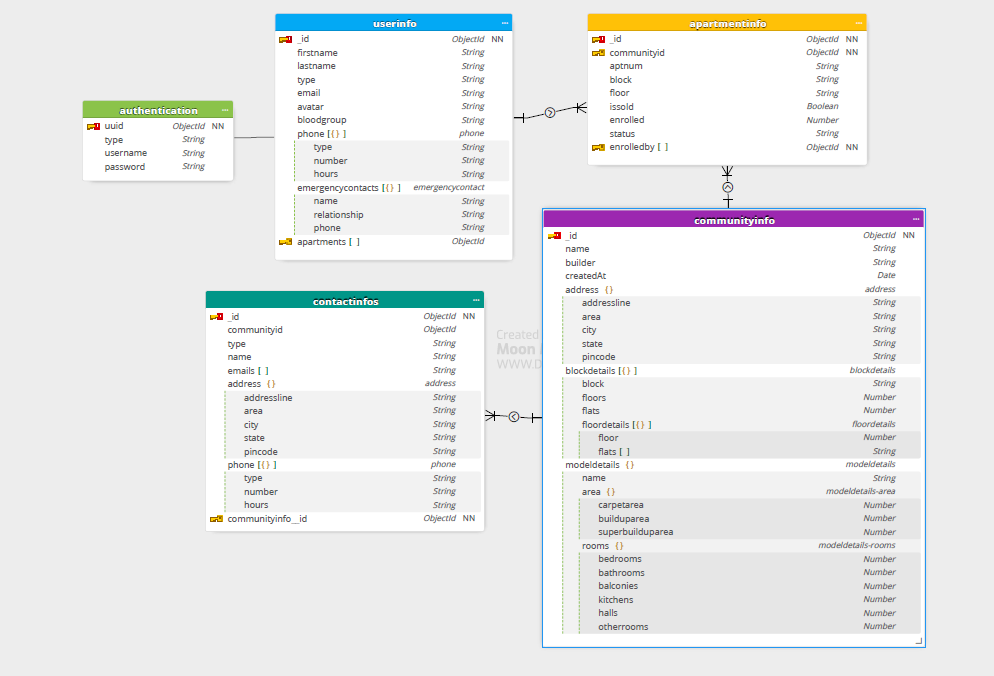
## **6.1 CommunityInfo & Apartment Info Collection:**

CommunityInfo and ApartmentInfo collection has complete details of apartments like , address, flats, floors, block, etc..



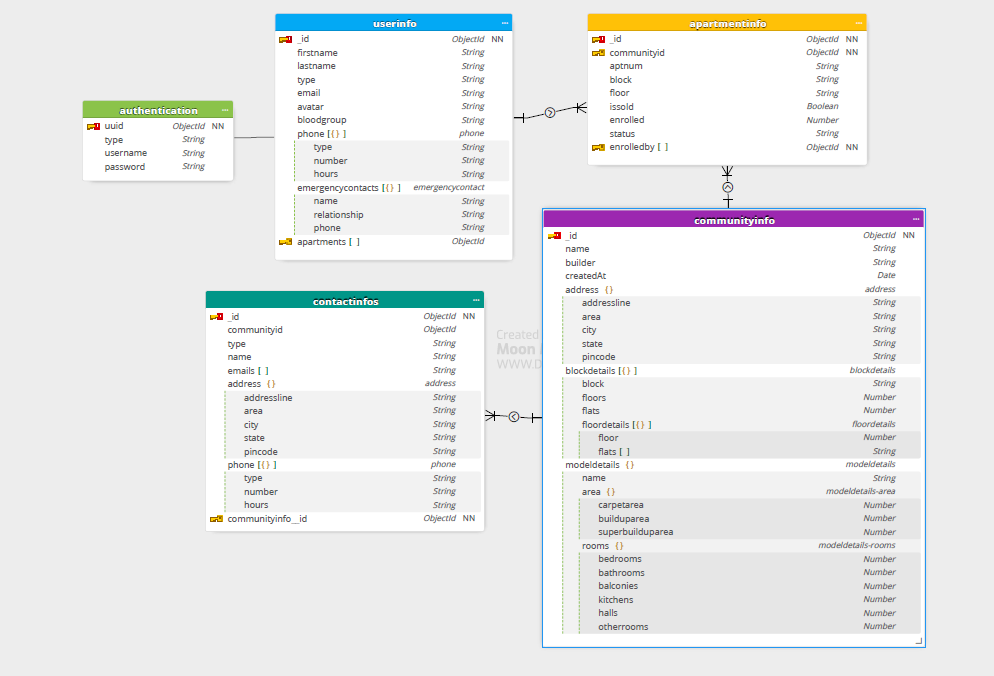
## **6.2 UserInfo Collection**

UserInfo collection holds resident and admin details (role, name, email, phone number, blood group, etc) except credentials details



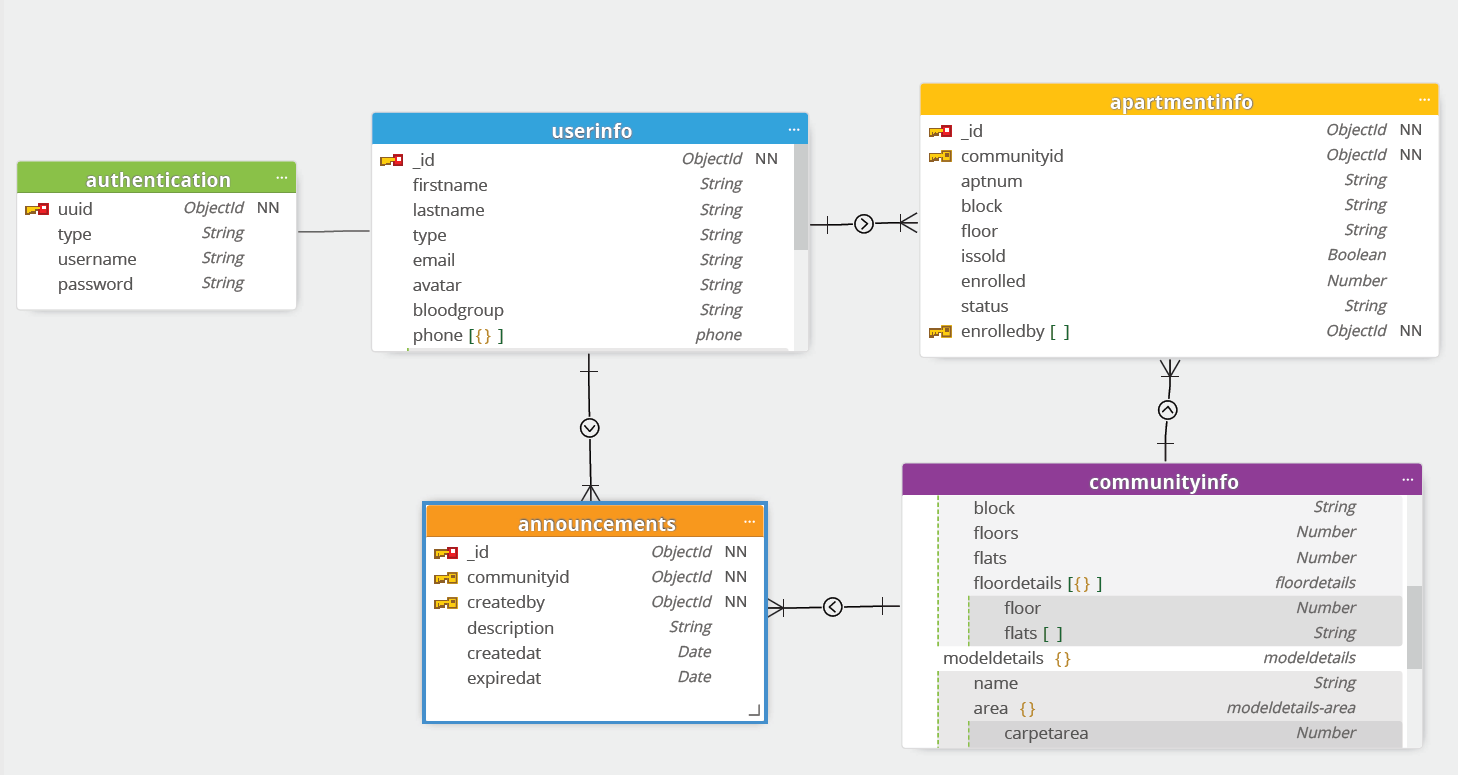
## **6.3 ContactInfo Collection:**

This service uses contactinfo collection. One document per community id. Community id itself uses id for this collection



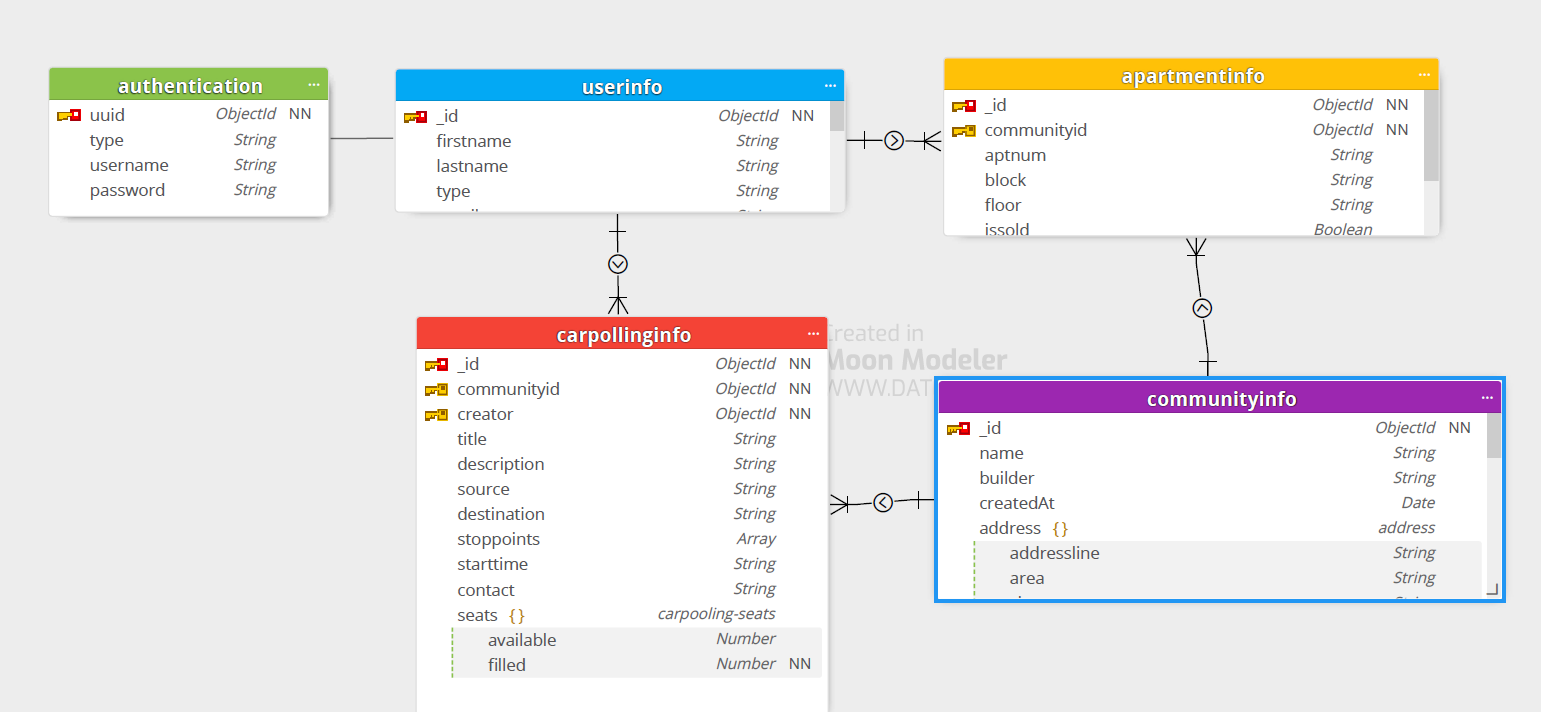
## **6.4 AnnouncementInfo Collection:**

Announcement service uses ‘announcementinfo’ collection. It has one to many relationships with userinfo and communityinfo. Multiple announcements can be made by one person and same goes for community.



## **6.5 CarpoolingInfo Collection**

Carpollinginfo collection used by carpooling service. It has one to many relationships with userinfo and communityinfo

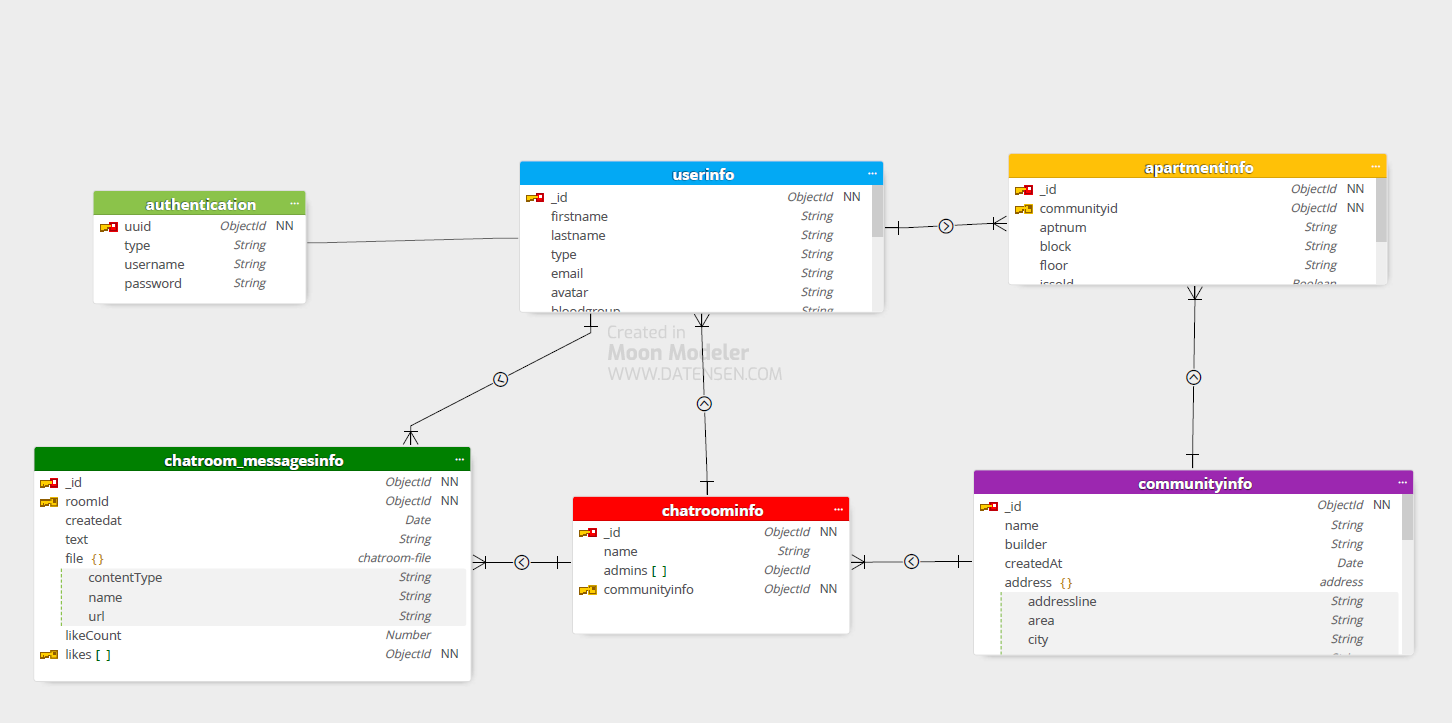


## **6.6 ChatRoomInfo Collection**

**Collections:**

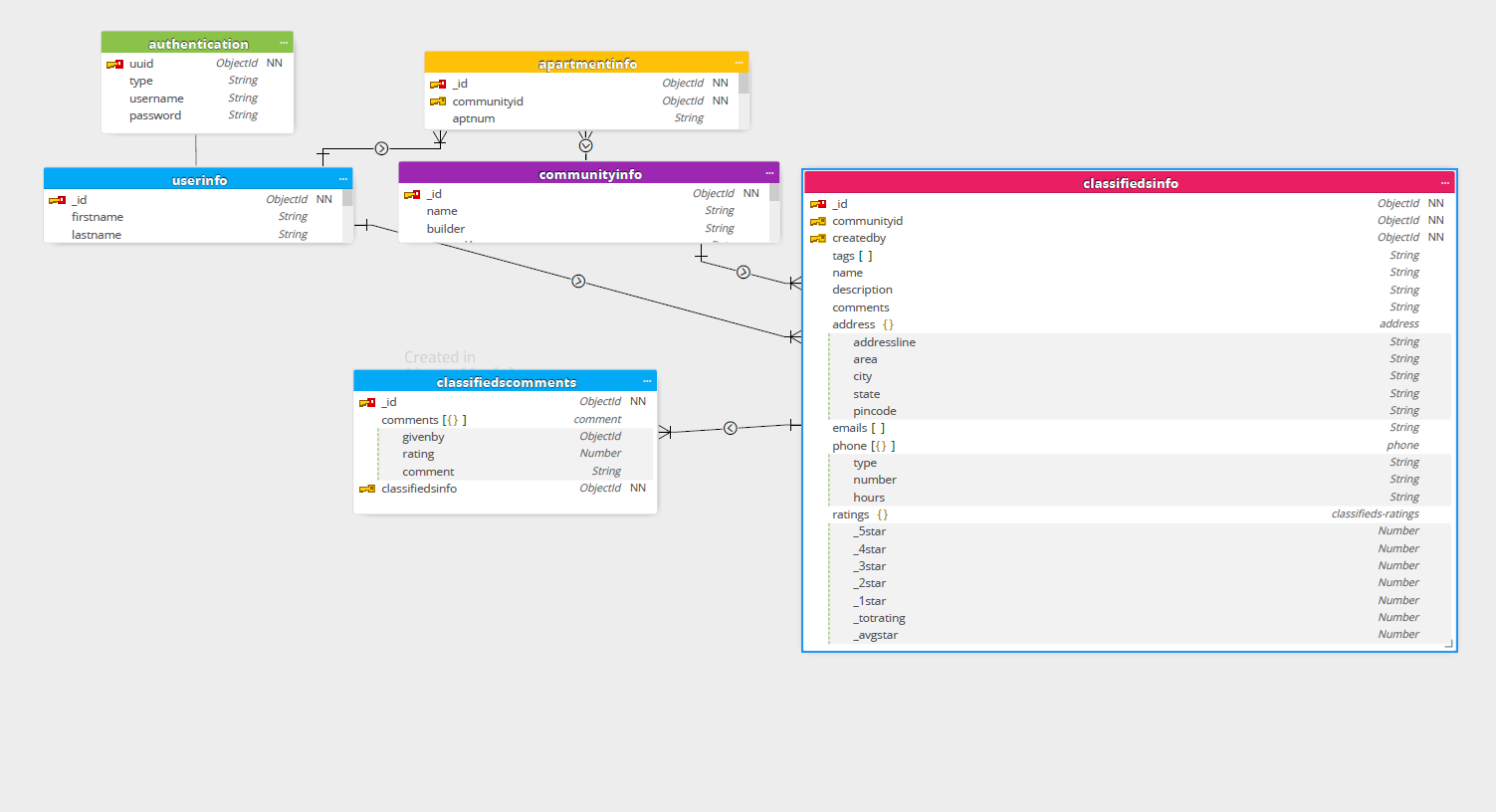
chatroominfo collection – stores chatroom details

chatroom\_messageinfo – stores each messages of the chat rooms

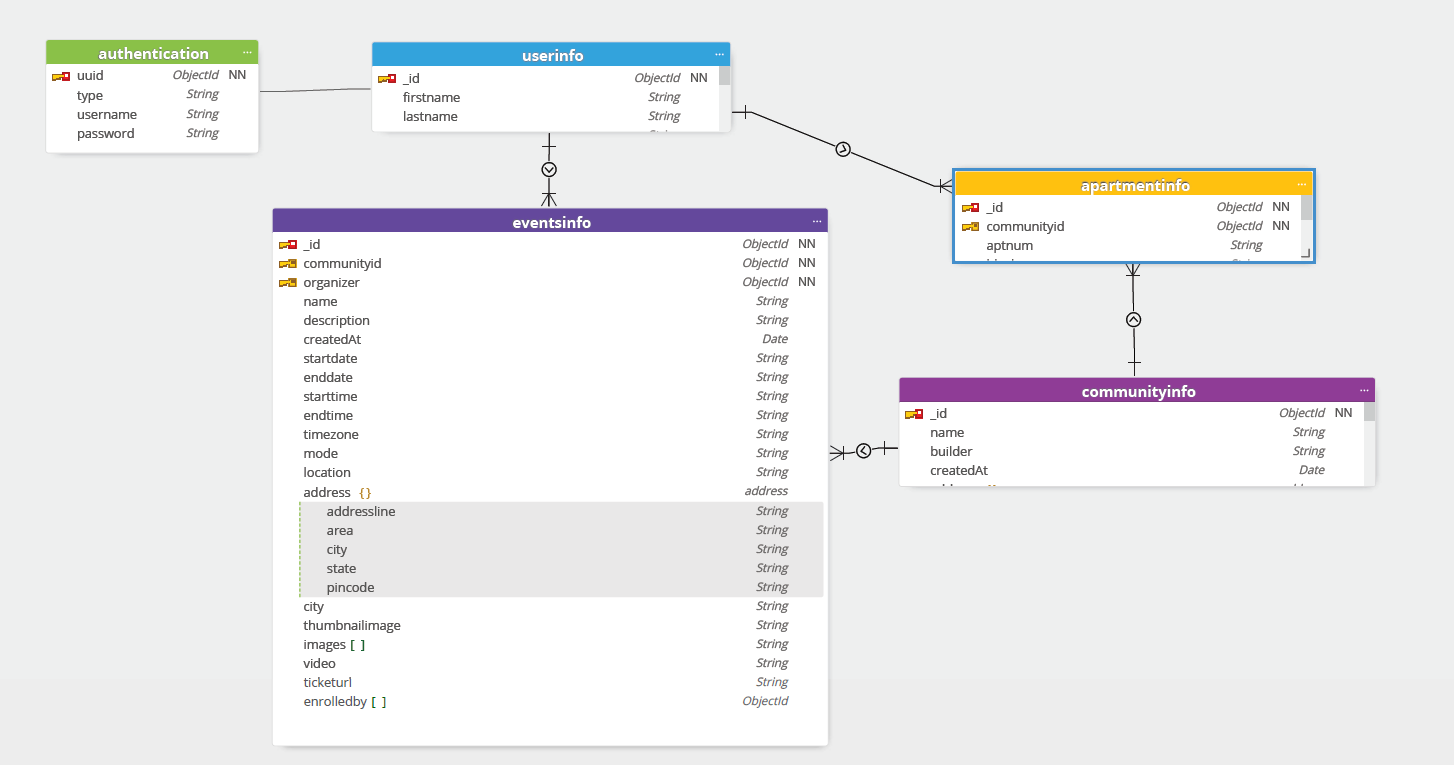


## **6.7 ClassifiedsInfo Collection**

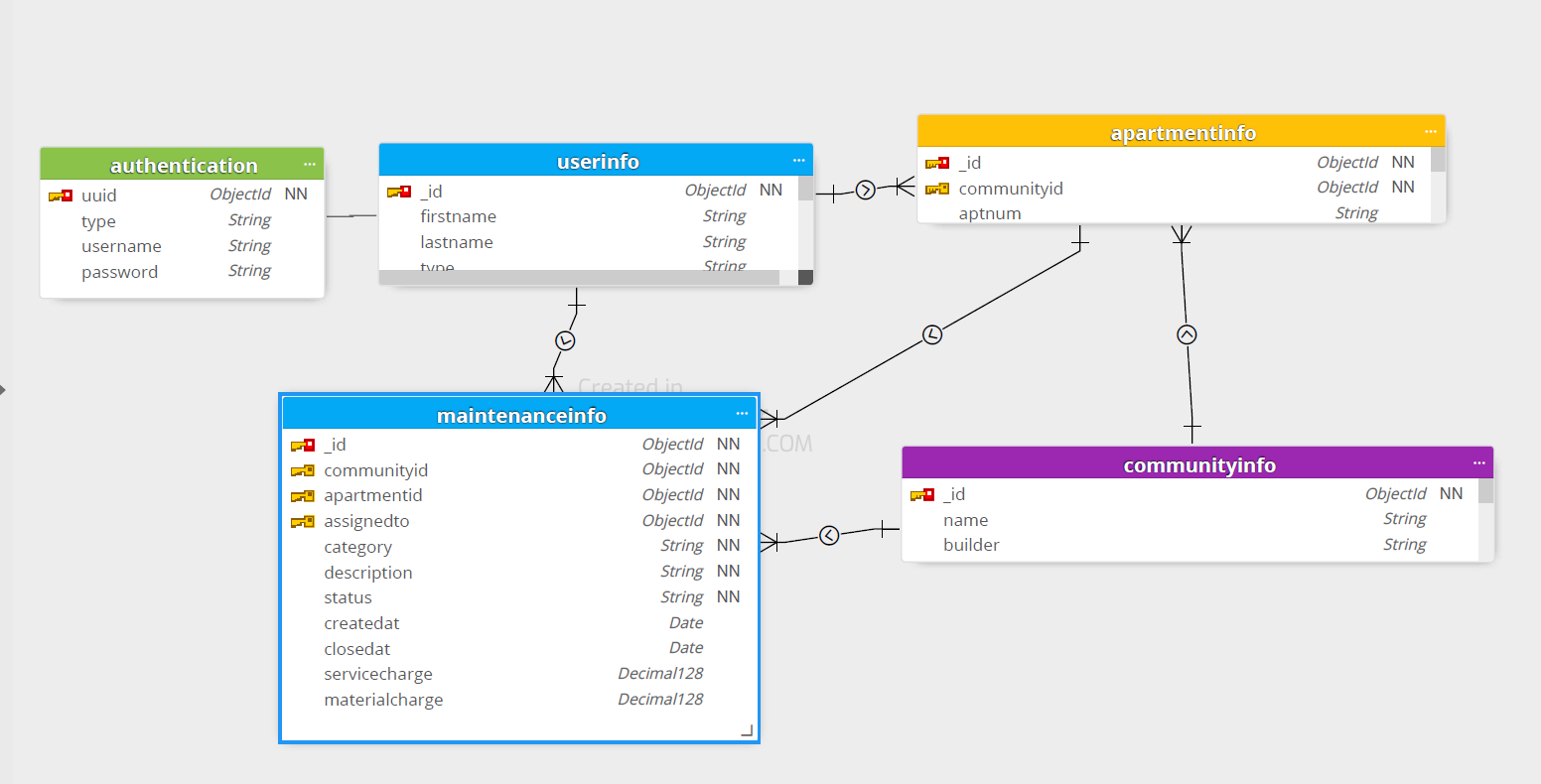
Classifiedsinfo and classifiedscomments collections are used by classifieds service. Classifiedscomments has details of each review comments provided for the classified.



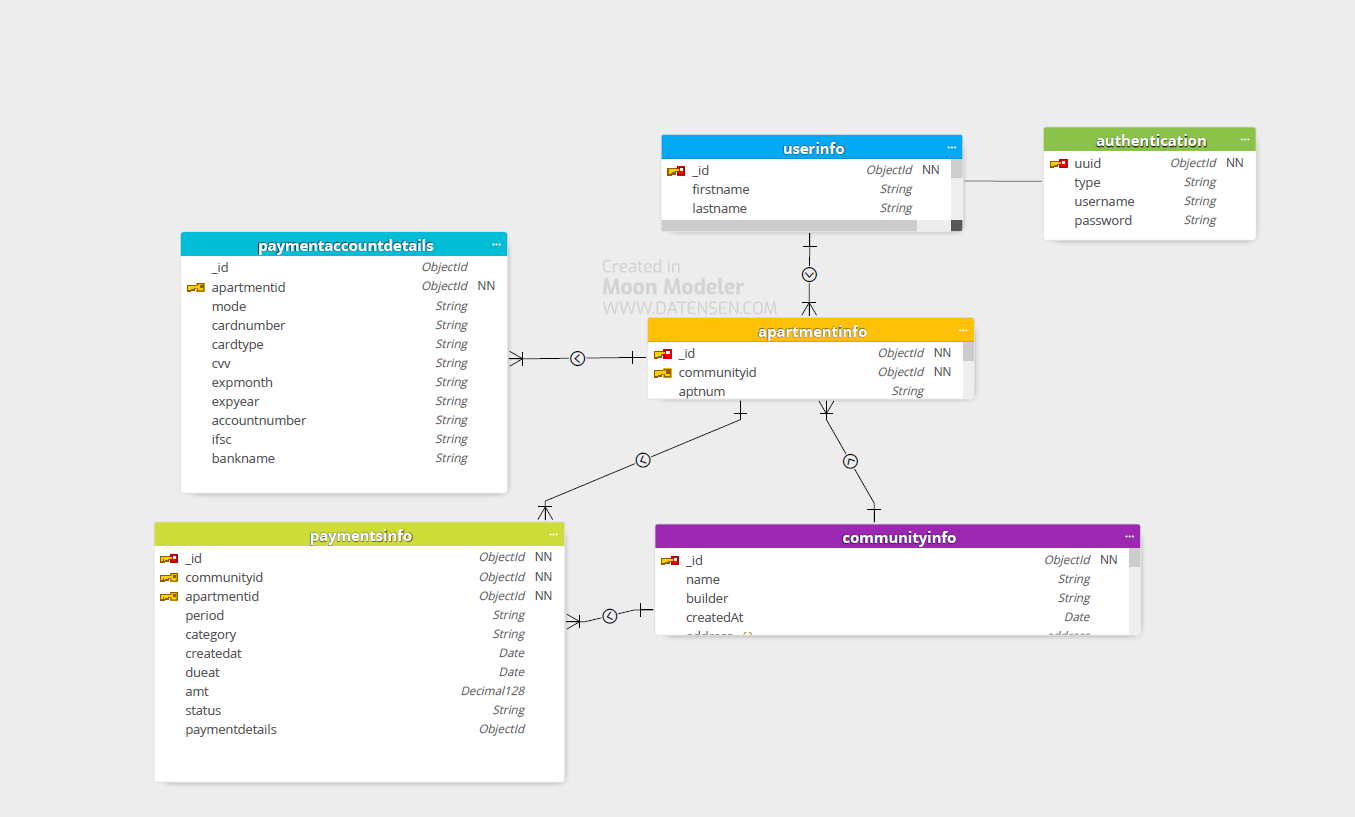
## **6.8. EventsmgmtInfo Collection**



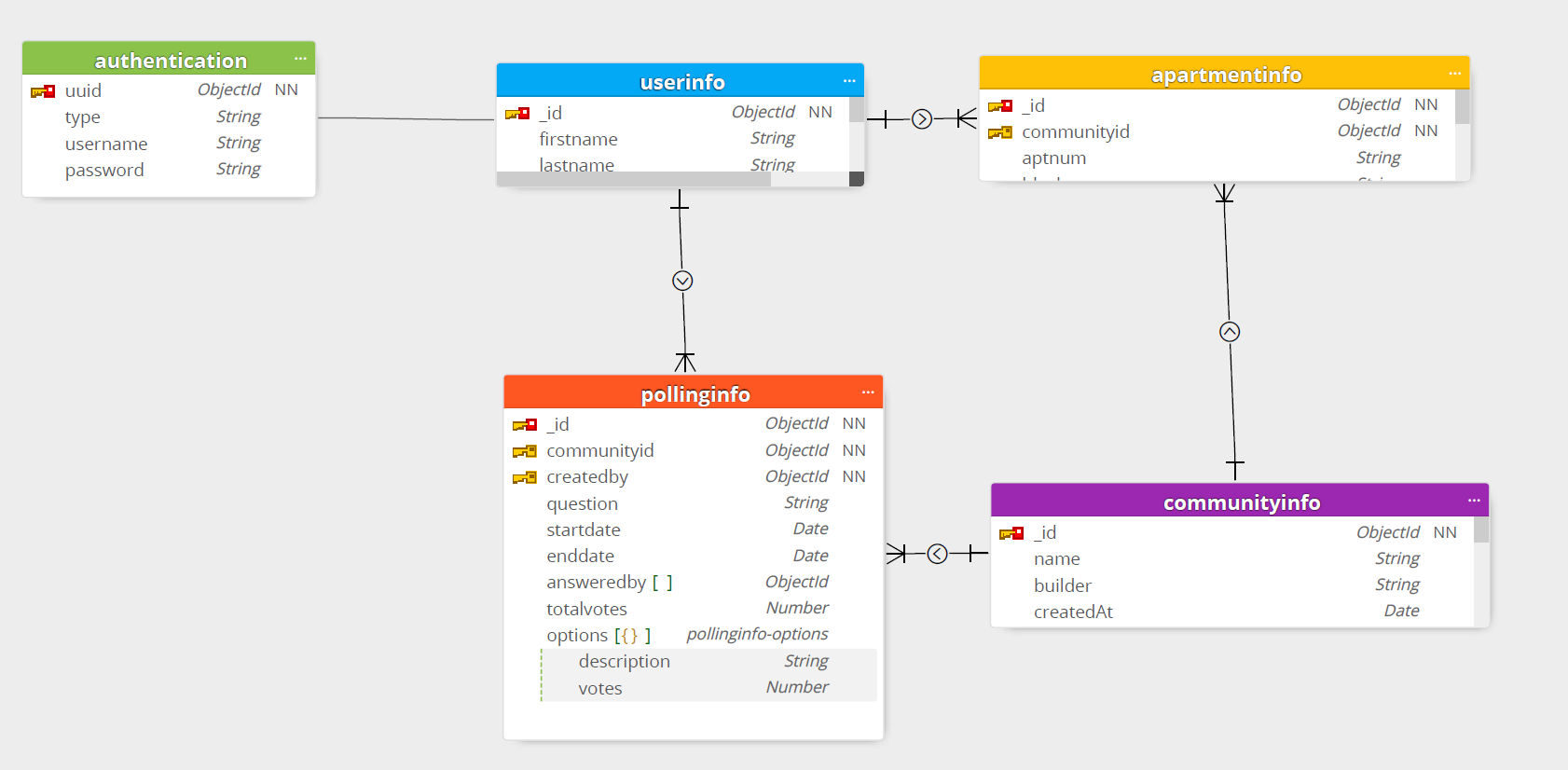
## **6.9 MaintenanceInfo Collection**



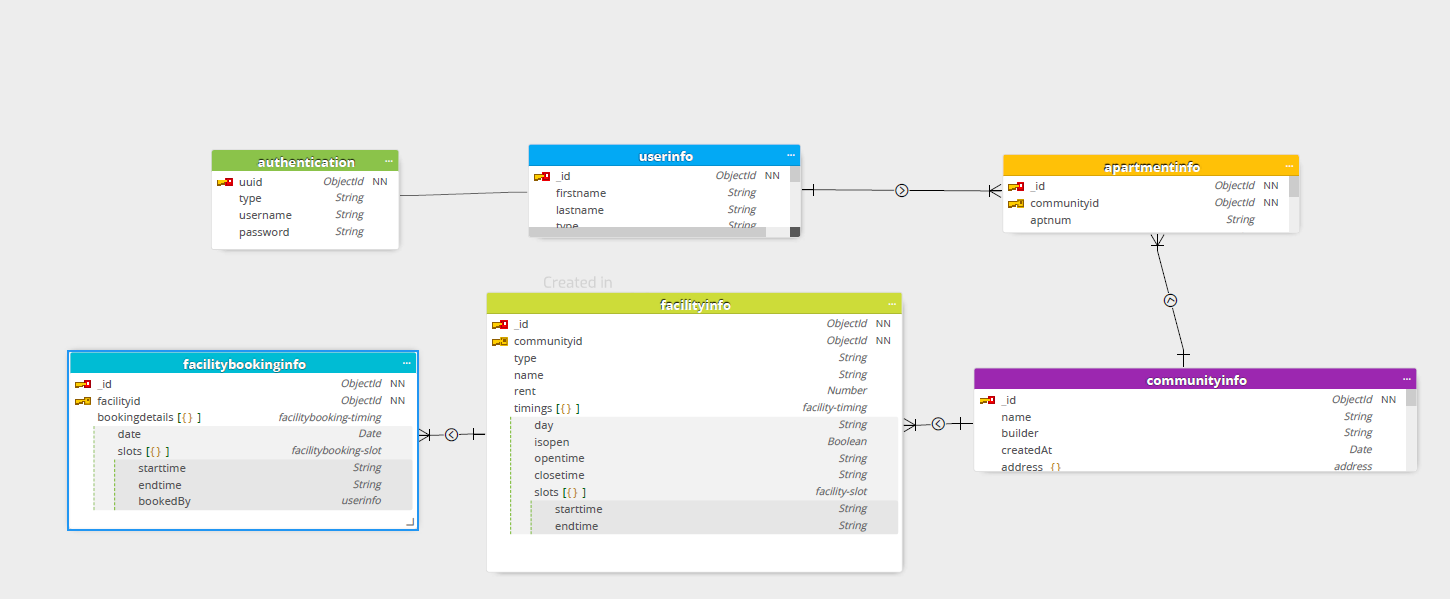
## **6.10. PaymentsInfo Collection**



## **6.11. PollingInfo Collection**



## **6.12. FacilityInfo Collection**



# **7. Prototyping**

Using Figma, we have created UI/UX design for our project. We are planning to implement the front end based on UI design shared in Figma

Refer UI designs of complete list of pages in below url (refer ‘ResidentConnect\_Desktop’) .

<https://www.figma.com/file/Z41Ggif8PV4WihLU1R0ALS/sample_project?node-id=1112%3A2219>

below is some samples UI design prepared,

