

מסמך ניתוח

תז: 322453689, 214695108, 214968240, 214662439
שם האפליקציה: Bettertogether

System Screens Structure You are required to present the full and clear structure of the system screens. The structure should include the following details:

1. List of Screens: o Provide a detailed list of all the screens in the system, such as: Home Page, User Profile Page, Settings Page, Search Results Page, etc.
2. Connections Between Screens: o Present the flow between the screens: how the user navigates from one screen to another (e.g., clicking a button on the "Home Page" leads to the "Profile Page").
3. Visual Representation: o Screens can be presented in a graphical format (e.g., diagram or screenshots). o If the system is already developed, include screenshots of the actual screens. o If the system is in the planning stage, provide a screen diagram created using software (e.g., PowerPoint or Word).

System Screens Structure

1. רשימת מסכים:

1. מסך כניסה:
 - o מאפשר התחברות באמצעות Google, Facebook, או אימייל וסיסמה.
2. מסך התראות:
 - o מציג עדכונים אישיים וגלובליים.
3. מסך תשלום:
 - o מבצע תשלומים בין המשתמש לאפליקציה ומחובר ל-API של PayPal.
4. מסך בית:
 - o מציג גישה מהירה לחדרי הימורים שהמשתמש משתמש בהם ומעבר למסך התראות.
5. מסך פרופיל:
 - o מציג פרטי משתמש, כולל היסטוריית הימורים, תמונת פרופיל, דירוג ואפשרות להוסיף חברים.
6. מסך יצירת חדר:
 - o טופס המאפשר יצירת חדר חדש שאליו משתמשים אחרים יכולים להצטרף.
7. מסך תפריט אישי של חדרים:
 - o מציג את החדרים שהמשתמש הצטרף אליהם או הימר בהם.
8. מסך חדר:
 - o מציג פרטים על החדר, כולל צ'אט, פרטי הימור ואפשרות להמר.

9. מסך חיפוש חדרים:
- מציג חדרים שפתחו משתמשים אחרים, כולל אפשרות חיפוש.
10. מסך חיפוש משתמשים:
- מציג משתמשים באפליקציה וכולל סרגל חיפוש למציאת משתמשים.
11. סרגל ניווט:
- מספק גישה מהירה:
 - למסך הבית.
 - למסך הפרופיל.
 - למסך חיפוש חדרים.
 - לתפריט האישי של חדרים.
-

2. חיבורים בין המסכים:

1. מסך כניסה → מסך בית:
- פעולה: התחברות לחשבון.
2. מסך בית:
- לחיצה על כפתור **תשלום** → מעבר למסך **תשלום**.
 - לחיצה על כפתור **התראות** → מעבר למסך **התראות**.
 - גישה לסרגל ניווט:
 - מסך בית.
 - מסך פרופיל.
 - מסך חיפוש חדרים.
 - מסך תפריט אישי של חדרים.
3. מסך **תשלום** → API של PayPal:
- פעולה: מעבר לממשק של PayPal.
4. מסך **התראות**:
- פעולה: אפשרות להיכנס להודעות.
5. מסך **תפריט אישי של חדרים** → מסך **חדר**:
- פעולה: לחיצה על חדר מסוים בתפריט.
6. מסך **חיפוש חדרים** → מסך **חדר**:
- פעולה: בחירת חדר לאחר חיפוש.
7. מסך **חיפוש משתמשים** → מסך **פרופיל**:
- פעולה: בחירת משתמש לאחר חיפוש.
8. מסך **פרופיל**:
- פעולה: אין ניווט נוסף (null).
9. מסך **חדר**:
- פעולה: אין ניווט נוסף (null).
-

TODO:

שקד תעשה את הUI

3. ייצוג ויזואלי:

Design Patterns for Android Studio

בהתבסס על מבנה הפרויקט שלנו, האפליקציה שלנו מיישמת ארכיטקטורה של MVVM (Model-View-ViewModel):

1. מבנה המחלקות וממשק הגרפי (Activities and Layouts):
 - מחלקות כמו `HomeActivity`, `LoginActivity`, `ProfileActivity` ו-`RoomsActivity` משמשות כ-View. הן אחראיות על אינטראקציה עם המשתמש והצגת הנתונים.
 - קבצי ה-XML בתיקיית `layout` מגדירים את ממשקי המשתמש (UI) של המחלקות הללו.
2. שימוש ב-Firebase:
 - העובדה שאנו משתמשים ב-Firebase לאחסון נתונים מצביעה על דפוס של `Data Source` או `Repository`, שהוא חלק חשוב בארכיטקטורת MVVM.
 - `Firebase` פועל כ-Model, מספק עדכוני נתונים בזמן אמת ואחסון מתמשך.
3. `Adapters` ו-`Utilities`:
 - מחלקות כמו `FirestoreUtils` מצביעות על ריכוז הטיפול באינטראקציות עם הנתונים, מה שמייצג את שכבת ה-Model.
 - `RoomsAdapter` ו-`ChatAdapter` משמשים כעוזרים לחיבור הנתונים ל-UI, שזה חלק ב-MVVM שמבצע הפרדה בין הנתונים לממשק המשתמש.
4. ניווט ומודולריות:
 - הפרדה של המחלקות והתיקיות השונות לפונקציות באפליקציה משקפת על הפרדה של אחריות, שהיא עיקרון מרכזי ב-MVVM.

1. שימוש ב-Model (Firebase)

- `Firebase` מספק ניהול נתונים בזמן אמת ושירותים כמו `Authentication`, `Firestore` ו-`Storage`, שמתאימים מאוד לשכבת ה-Model בארכיטקטורת MVVM.
- `Firebase` מאפשר עדכון נתונים אוטומטי ואסינכרוני, דבר שמקל על הפרדה בין הלוגיקה העסקית (Model) לממשק המשתמש (View).

2. הפרדה ברורה של אחריות

- ב-MVVM כל שכבה אחראית לתפקיד ייחודי:
 - `Model`: אחראי על הנתונים, כולל קריאות ל-Firebase, ניהול נתונים מקומי.
 - `View`: אחראית על הצגת הנתונים למשתמש והאינטראקציה עם המשתמש (מחלקות וקבצי XML).
 - `ViewModel`: מתווך בין ה-Model ל-View. זה מאפשר לנהל את הלוגיקה מבלי שה-View תלוי בנתונים ישירות.

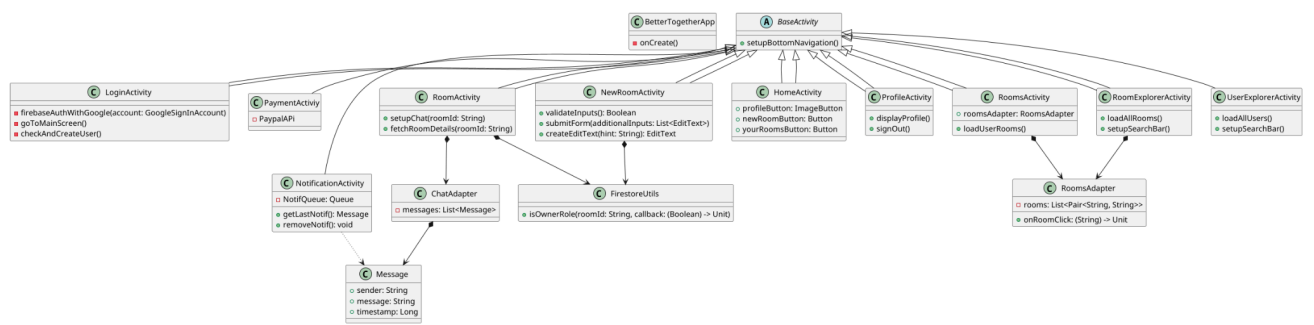
3. קלות בניהול עדכוני UI בזמן אמת

- **Firestore** מעדכן נתונים בזמן אמת. ב-MVVM, העדכונים האלה יכולים לעבור ישירות ל-View דרך ה-ViewModel.
 - זה חוסך את הצורך לכתוב קוד מורכב ב-Activity, ומאפשר עדכון אוטומטי של ה-UI כאשר הנתונים משתנים.
-

למה לא MVP או MVC?

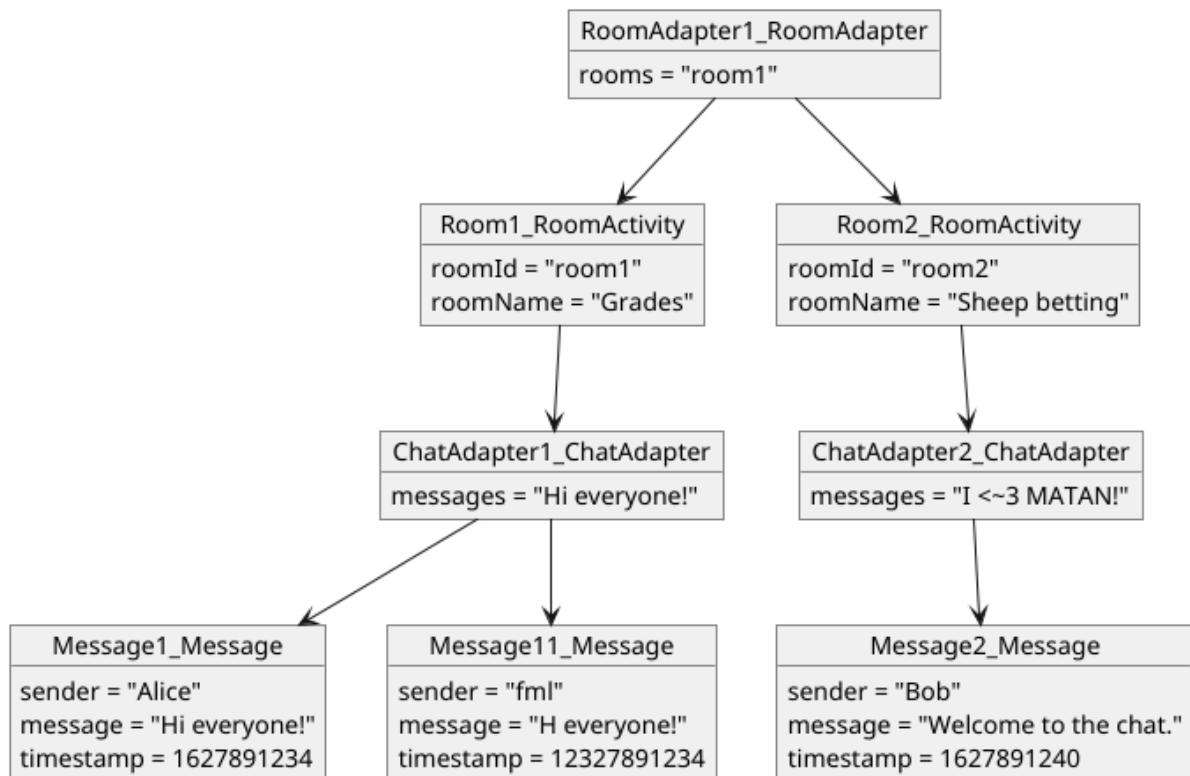
- **MVC**: מבנה פשוט יותר, אבל הוא לא מתאים לאפליקציות מודרניות עם עדכוני נתונים מורכבים. ה-Controller (או ה-Activity) הופך לעמוס מאוד.
- **MVP**: מתאים לפרויקטים קטנים, אבל כאשר האפליקציה גדלה, ה-Presenter עלול להפוך למורכב מדי, במיוחד עם **Firestore**.
- **MVVM** מאפשר שימוש בכלים מודרניים כמו **Data Binding** ו-**LiveData**, ומפשט עבודה עם נתונים בזמן אמת, דבר שמייחד אותו במיוחד לאפליקציות מבוססות **Firestore**.

Class Diagram

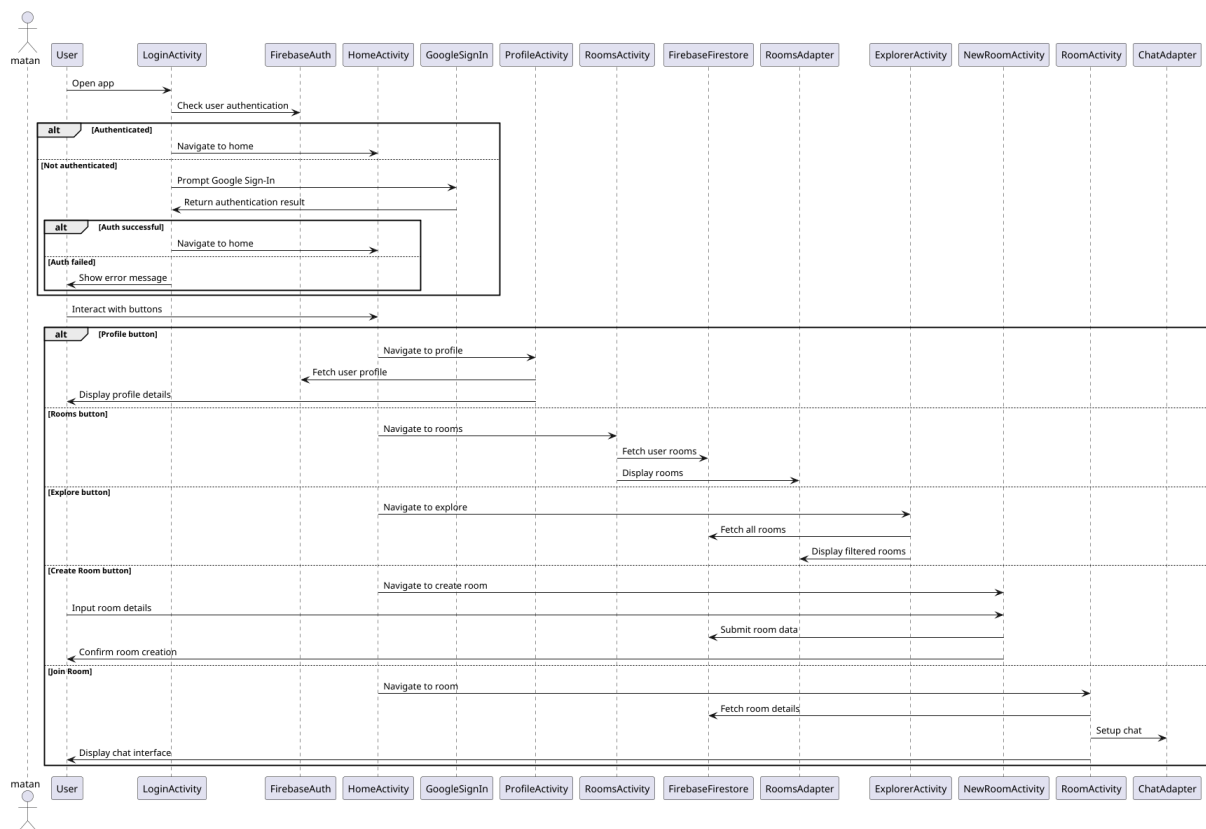
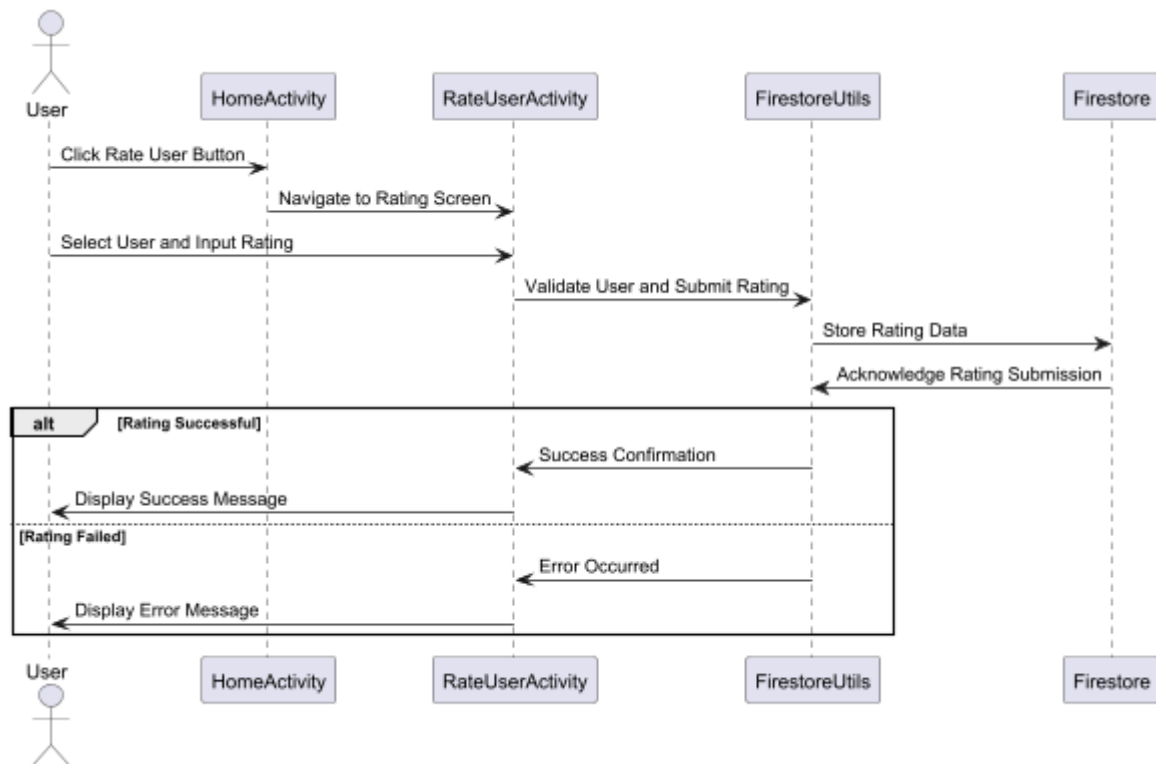


Object Diagram

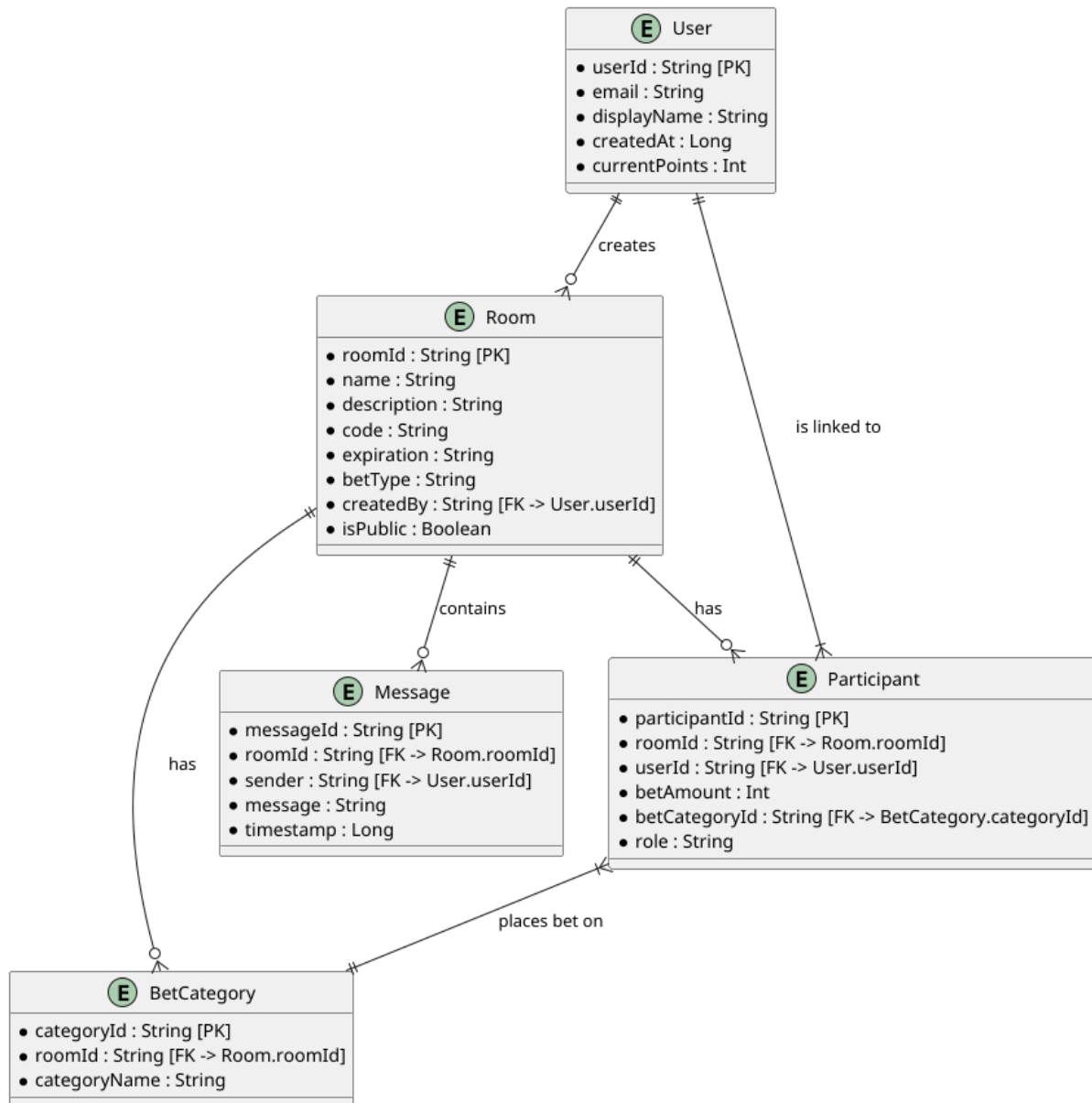
באשר התו '_' מסמן את התו ':' (על מנת להראות שייכות של עצם למחלקה)



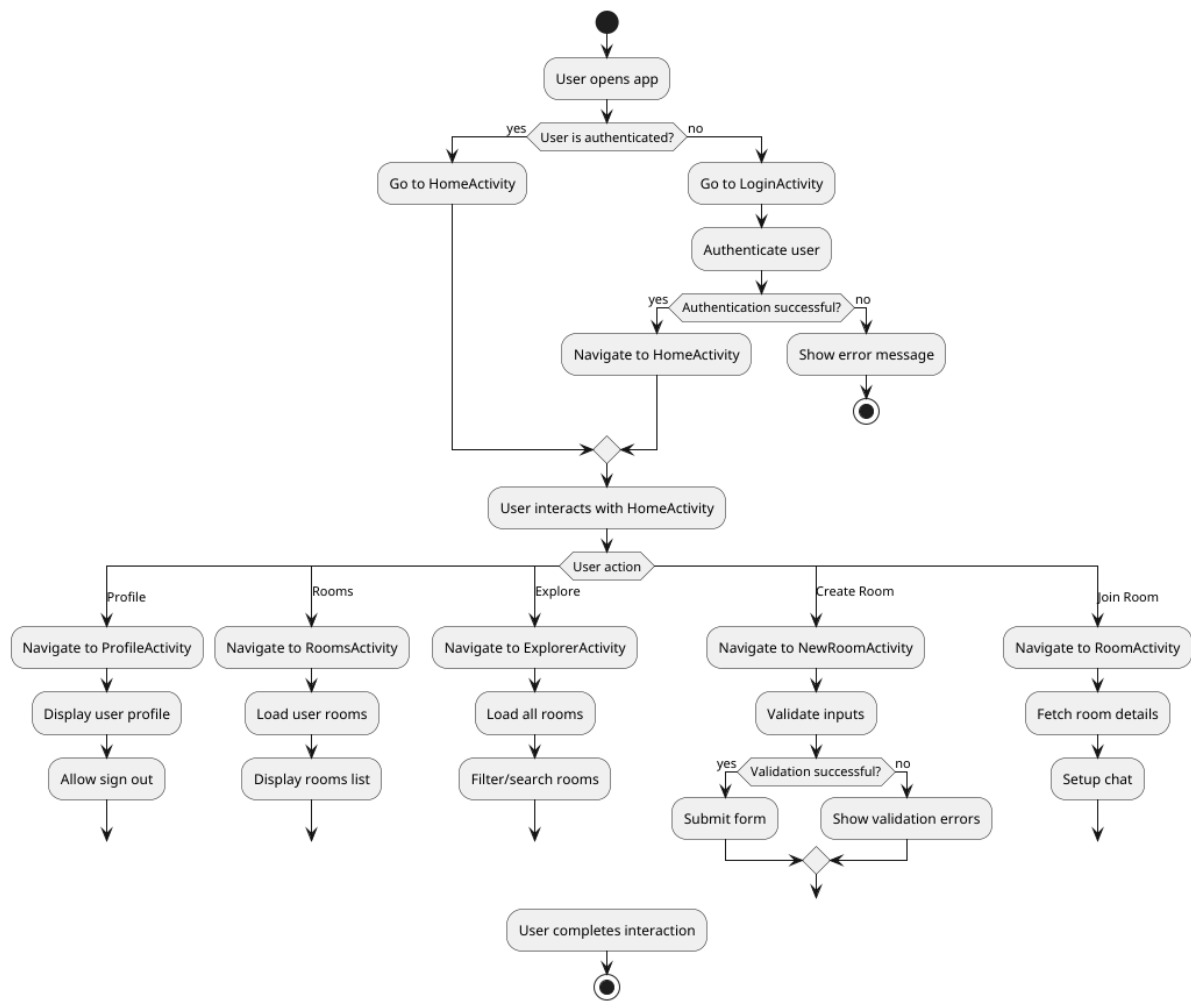
Sequence Diagram



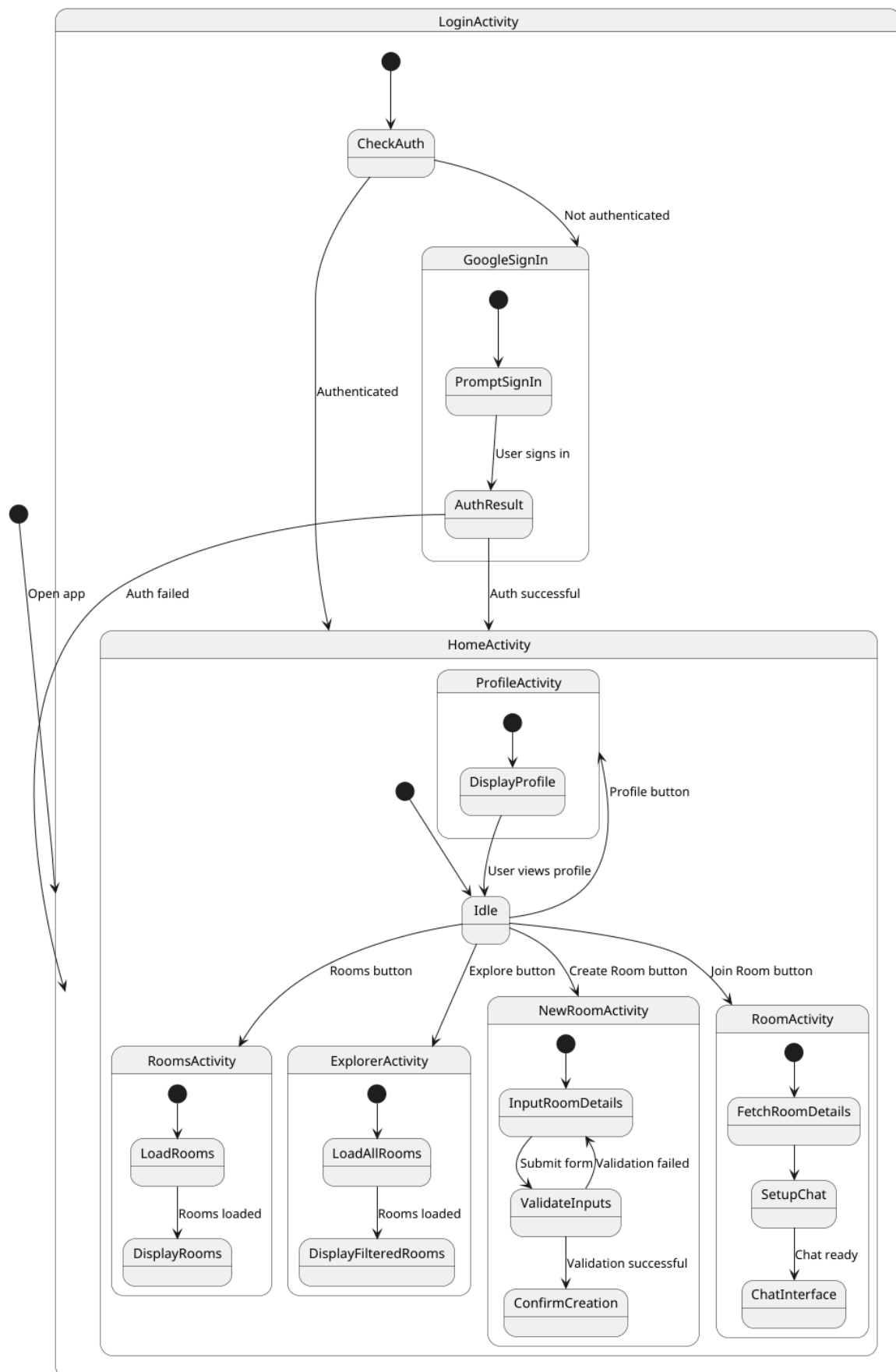
ERD



Activity Diagram

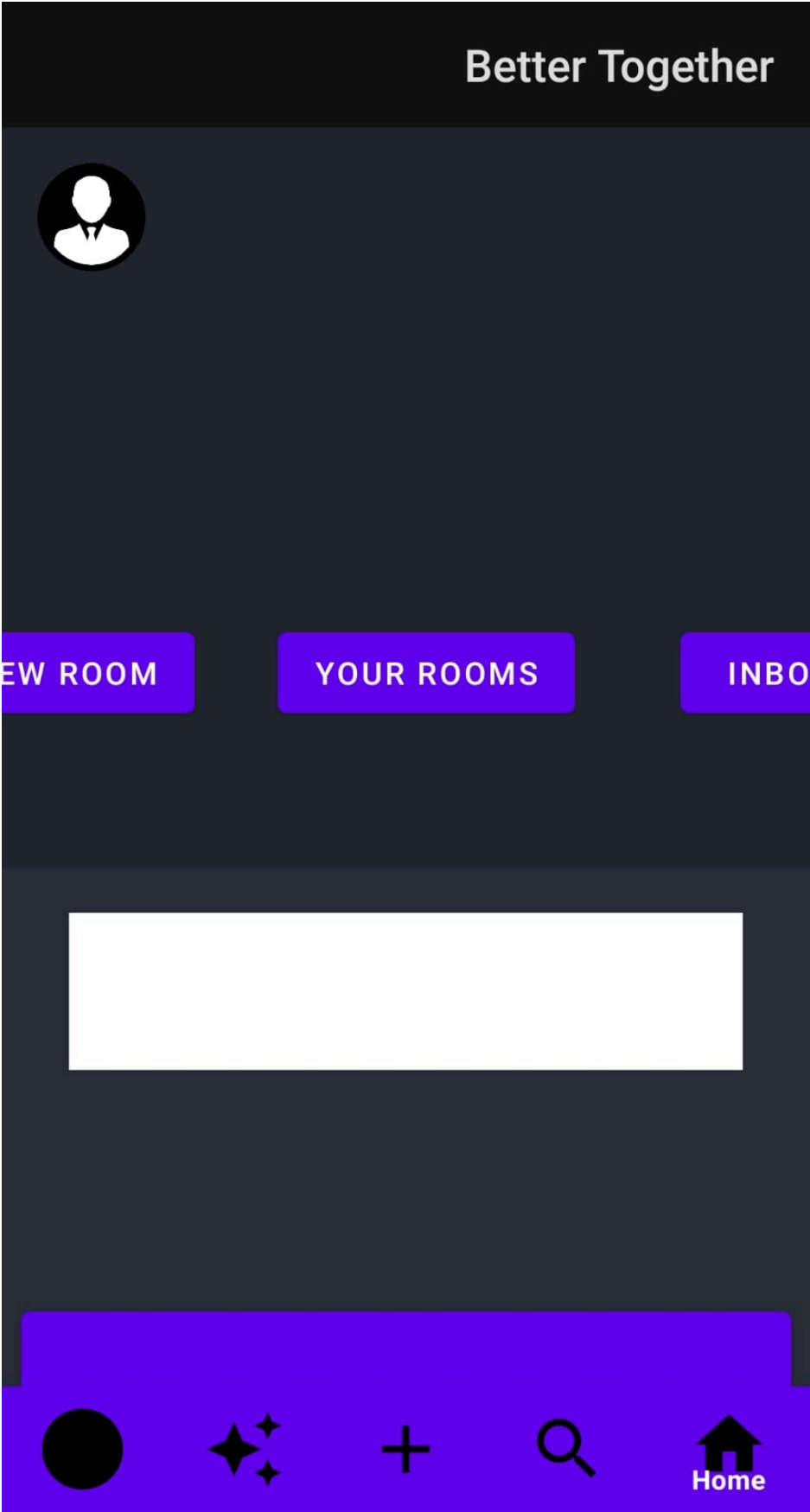


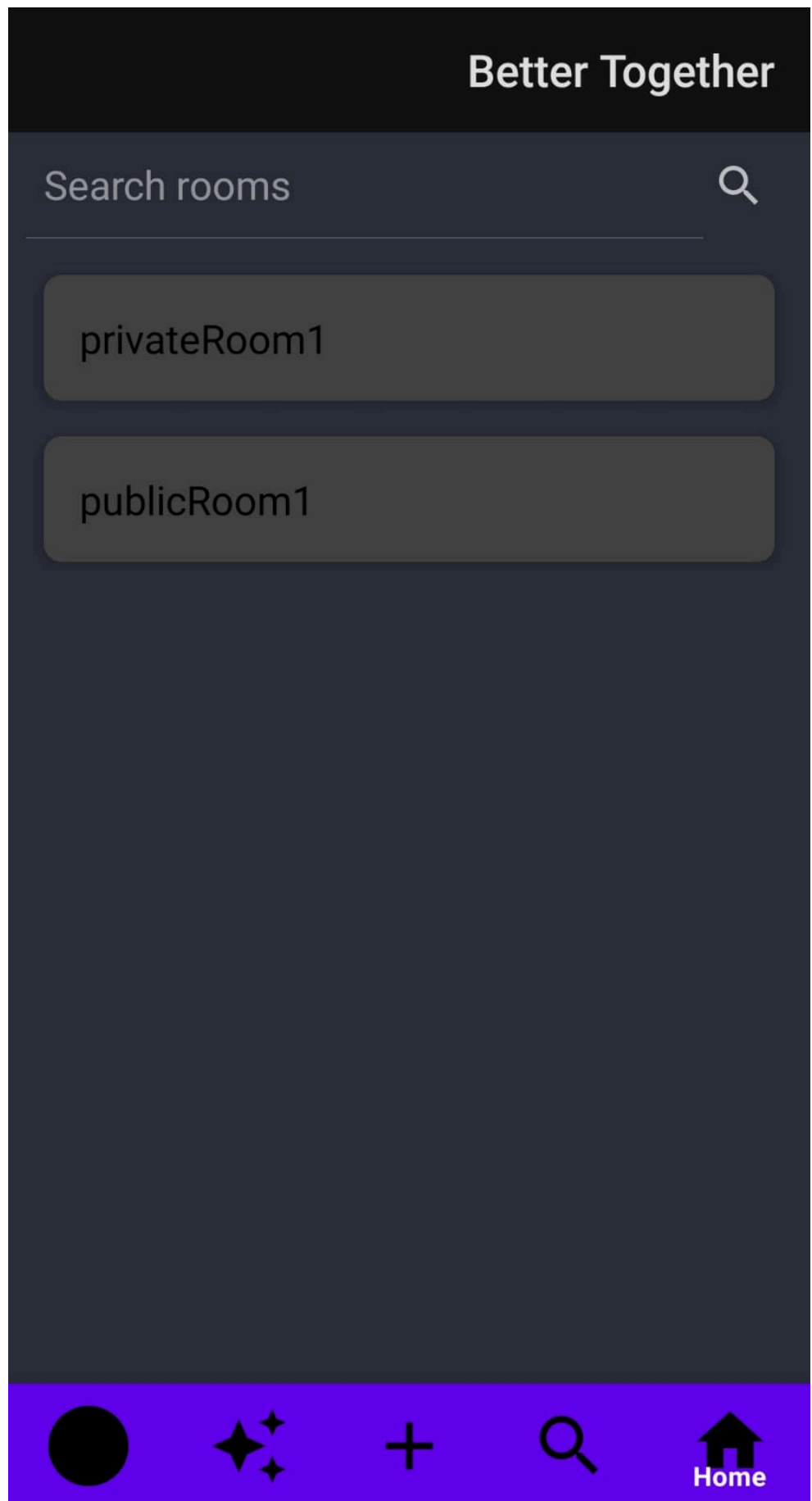
State machine



מסכים באפליקציה:

מסך בית





Better Together

Bet Subject

Bet Number

Expiration Date

Description

Room Code

Public Room ☐

Event? ☐

Even Bet ☐


Ratio Bet ☐

SUBMIT

+

Home

Better Together



Roi Siboni

About

This is the bio section.

User Information

N/A

Username

N/A

Gender

N/A

Age

N/A

Date of Birth



N/A

Mobile






roisi20041@gmail.com

Email

Social Media

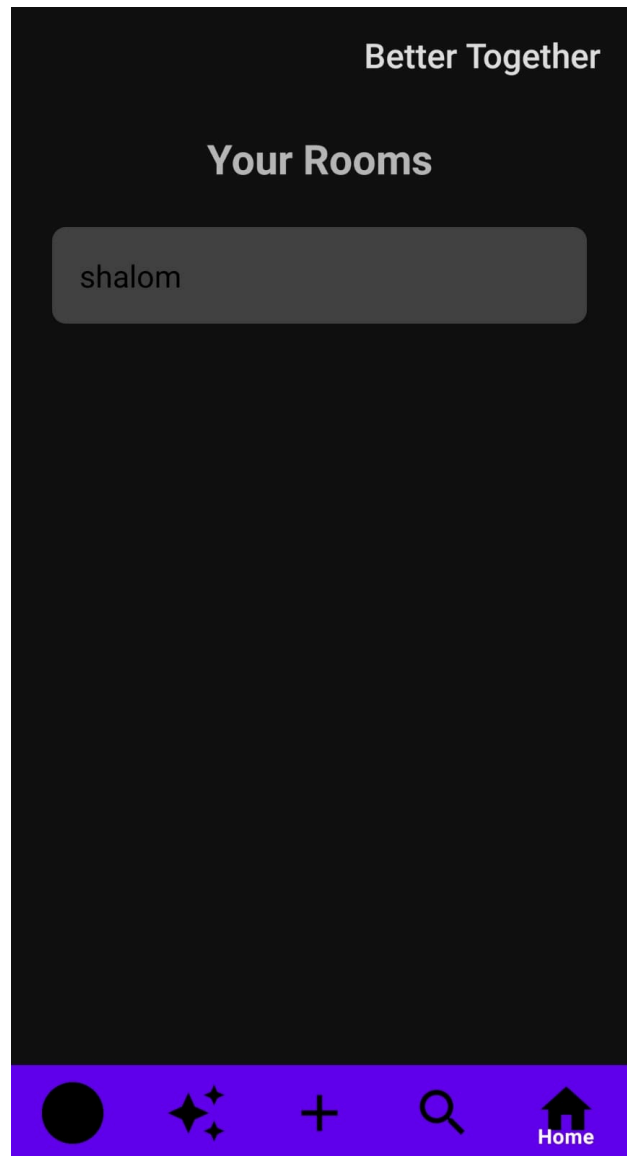


SIGN OUT

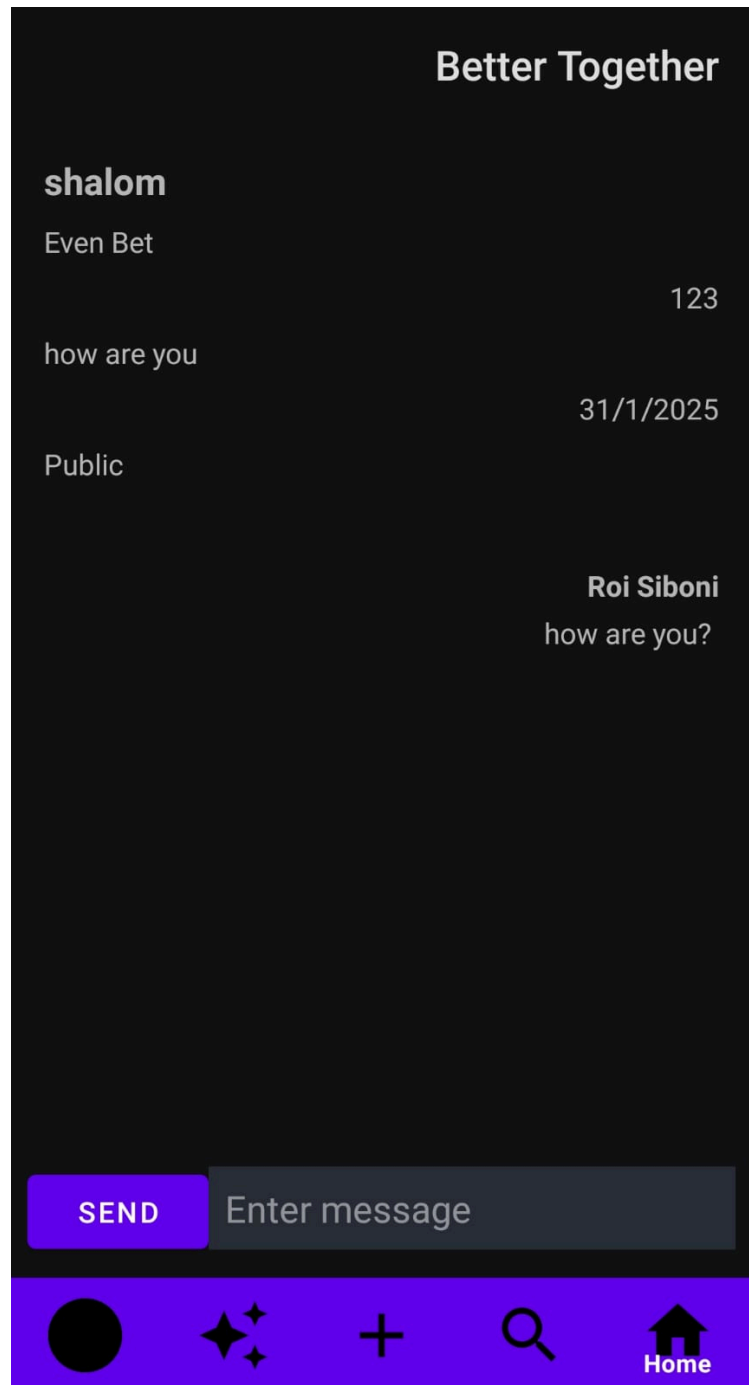


Home

חדר החדרים שהמשתמש חלק מהם:



מסך החדר עצמו כולל צאט של החדר:



מסך לוגין:
(יש לנו עוד להוסיף כניסה בעזרת שם משתמש וסיסמה ופייסבוק אבל זה הבסיס לזו)
הגדרנו בהתאם לשקיעה והזריחה שהמסך כניסה ישתנה (:

