DAMG 6210 - Data Management and Database Design

Roommate and Accommodation Management System

Group Number 9

Adarsh Suryavanshi - 002894722
Ashwatha E. S - 002308776
Pratik Kanade - 002311591
Priyank Bagad - 002338822
Sarthak Sargar - 002057385

Project Overview

ROOM & MATES is a modern web application oriented to change the one-sided paradigm of how people search for a place to live and find compatible roommates.

Designed with diversity in mind, it makes searching for housing options and cohabitants that fit your preferences both easy and fast.

The system is designed to especially meet challenges faced by people moving to new cities or regions, be it international students, young professionals, or people moving for work or education.

ROOM & MATES seeks to fill that gap by connecting users with resources and people who will make their transition easier and more convenient.

The system includes important features such as tailored matchmaking from user inputs, through which user preferences—-budget, location, lifestyle, and personality traits—are taken into consideration.

This new approach eases logistical issues while simultaneously building a sense of community building and exchange of culture.

Whether student or professional, adventurer or dreamer, ROOM & MATES secures your start within just the right home-away-from-home environment.

Purpose

The primary purpose of ROOM & MATES is to revolutionize and upgrade the process of finding accommodation and preferred roommates by :

1. Simplifying the search process for both rooms and compatible cohabitants.

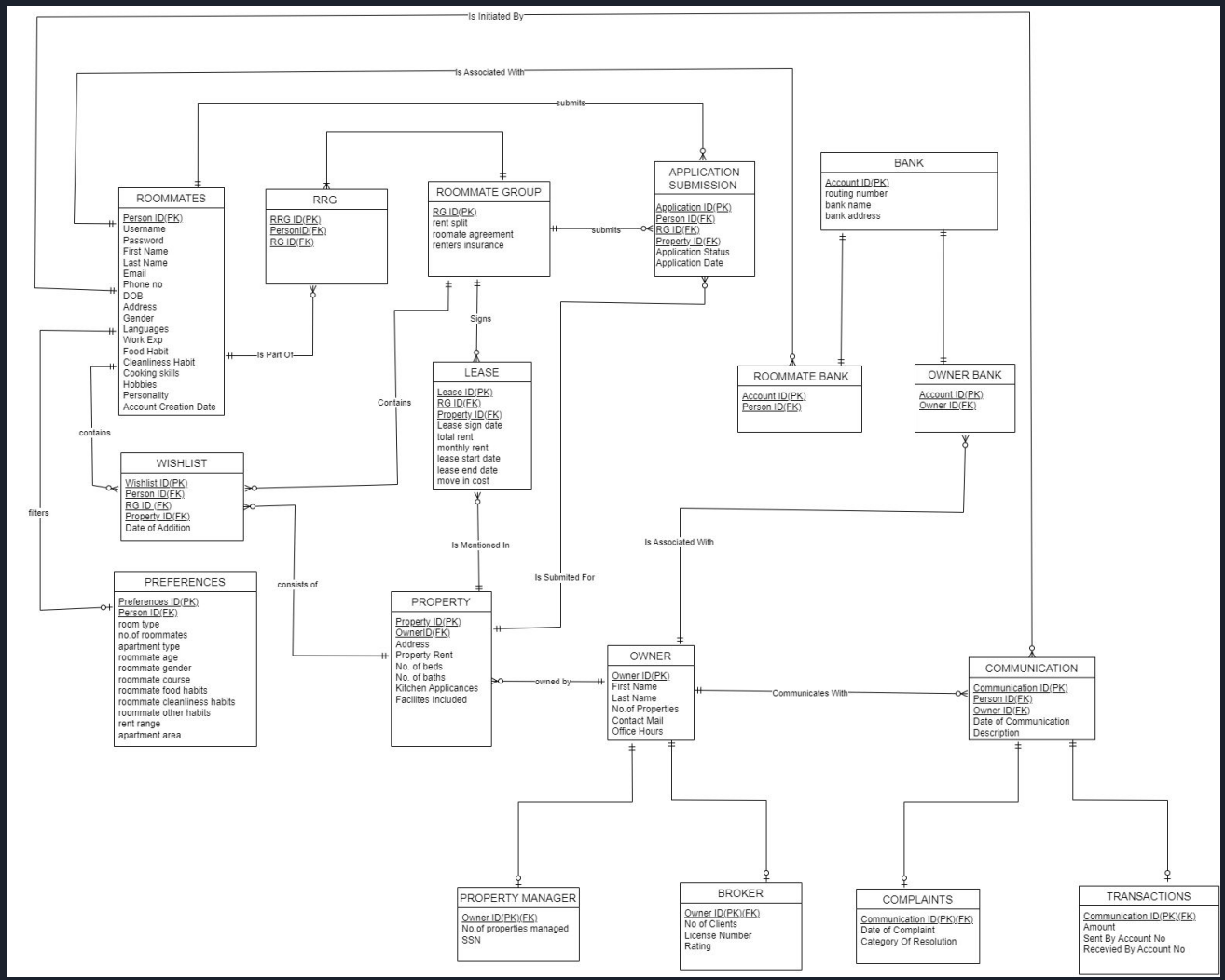
2. Reducing the time and effort required to find suitable living arrangements for available locations.

3. Finding accommodation with preferable roommates with each student's chosen filters.

4. Providing a secure and reliable platform for all aspects of the room search and roommate-matching process.

5. Ultimately, creating a more efficient, reliable and user-friendly solution to address the challenges of finding housing in desired locations and building connections in new environments.

Entity - Relationship Diagram



Stored Procedures

```sql
-- Get the property details by sepcifying the type of property and the number of bedrooms and bathrooms

CREATE PROCEDURE GetPropertyDetails
    @PropertyType VARCHAR(50),
    @Beds INT,
    @Baths INT
AS
BEGIN
    SELECT p.PropertyID, p.Address, p.KitchenAppliances, p.FacilitiesIncluded,
           o.FirstName, o.LastName, o.ContactMail,
           l.MonthlyRent AS CurrentMonthlyRent, l.MoveInCost
    FROM Property p
    JOIN Owner o ON o.OwnerID = p.OwnerID
    JOIN Lease l ON l.PropertyID = p.PropertyID
    WHERE p.PropertyType = @PropertyType AND p.NoOfBeds = @Beds AND p.NoOfBaths = @Baths
END
GO
```

```sql
-- Get Flatmate details by mentioning preferences as input

CREATE PROCEDURE GetFlatmates
    @Gender VARCHAR(10),
    @FoodHabit VARCHAR(50),
    @CleanlinessHabit VARCHAR(50),
    @RoomType VARCHAR(50),
    @RentStart INT,
    @RentEnd INT
AS
BEGIN
    SELECT r.PersonID, r.FirstName, r.LastName, r.Email, r.Phone, r.Gender, r.Languages, r.WorkExp, r.FoodHabit,
           r.CleanlinessHabit, r.CookingSkills, r.Hobbies, r.Personality,
           p.RoomType, p.NoOfRoommates, p.ApartmentType, p.RentRange AS PreferredRent, p.RoommateAge AS PreferredRoommateAge,
           p.RoommateCourse AS PreferredRoommateCourse, p.RoommateGender AS PreferredRoommateGender,
           p.RoommateFoodHabits AS PreferredRoommateFoodHabits,
           p.RoommateCleanlinessHabits AS PreferredRoommateCleanlinessHabits,
           p.RoommateOtherDetails AS PreferredRoommateOtherDetails
    FROM Roommates r
    JOIN Preferences p ON p.PersonID = r.PersonID
    WHERE r.Gender = @Gender AND r.FoodHabit = @FoodHabit AND r.CleanlinessHabit = @CleanlinessHabit
          AND p.RoomType = @RoomType AND p.RentRange BETWEEN @RentStart AND @RentEnd
END
GO
```

User Defined Functions

```sql
-- 4. Calculate compatibility Score Between Two Roommate
CREATE FUNCTION dbo.CalculateCompatibility(…
)
RETURNS DECIMAL(5, 2)
AS
BEGIN
    DECLARE @Score DECIMAL(5, 2) = 0;

    -- Increment score for matching preferences
    SELECT @Score = @Score + 25
    FROM Preferences p1
    INNER JOIN Preferences p2 ON p1.PersonID = @PersonID1 AND p2.PersonID = @PersonID2
    WHERE p1.ApartmentType = p2.ApartmentType;

    SELECT @Score = @Score + 25
    FROM Preferences p1
    INNER JOIN Preferences p2 ON p1.PersonID = @PersonID1 AND p2.PersonID = @PersonID2
    WHERE p1.RoommateCleanlinessHabits = p2.RoommateCleanlinessHabits;

    SELECT @Score = @Score + 25
    FROM Preferences p1
    INNER JOIN Preferences p2 ON p1.PersonID = @PersonID1 AND p2.PersonID = @PersonID2
    WHERE p1.RoommateFoodHabits = p2.RoommateFoodHabits;

    SELECT @Score = @Score + 25
    FROM Preferences r1
    INNER JOIN Preferences r2 ON r1.PersonID = @PersonID1 AND r2.PersonID = @PersonID2
    WHERE r1.RoommateGender = r2.RoommateGender;

    RETURN @Score;
END;
GO
```

```sql
-- 3. Check Lease Overlap for Roommate Group
CREATE FUNCTION dbo.CheckLeaseOverlap(@RGID INT)
RETURNS BIT
AS
BEGIN
    DECLARE @OverlapCount INT;

    -- Count overlapping leases
    SELECT @OverlapCount = COUNT(*)
    FROM Lease l1
    CROSS JOIN Lease l2
    WHERE l1.RGID = @RGID AND l2.RGID = @RGID
      AND l1.LeaseID <> l2.LeaseID -- Exclude self-comparison
      AND l1.LeaseEndDate > l2.LeaseStartDate
      AND l1.LeaseStartDate < l2.LeaseEndDate;

    -- Return 1 if overlap exists, otherwise 0
    RETURN CASE
        WHEN @OverlapCount > 0 THEN 1
        ELSE 0
    END;
END;
GO
```

DML Triggers

```
-- DML trigger to store the data of properties that were removed from the wishlist

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='WishlistDeletedProperties' AND xtype='U')
BEGIN
CREATE TABLE WishlistDeletedProperties (
    DeletedPropertyID INT IDENTITY(1,1) PRIMARY KEY,
    WishlistID INT,
    PersonID INT,
    RGID INT,
    PropertyID INT,
    DateOfAddition DATE,
    DateofDeletion DATETIME DEFAULT GETDATE()
)
END
GO

CREATE TRIGGER DeletedPropertiesFromWishlist
ON Wishlist
AFTER DELETE
AS
BEGIN
    INSERT INTO WishlistDeletedProperties (WishlistID, PersonID, RGID, PropertyID, DateOfAddition, DateofDeletion)
    SELECT WishlistID, PersonID, RGID, PropertyID, DateOfAddition, GETDATE()
    FROM deleted
END
GO
```

```
-- DML Trigger to store the data of applications that were withdrawn

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='ApplicationWithdrawals' AND xtype='U')
BEGIN
CREATE TABLE ApplicationWithdrawals (
    AppWithdrawalID INT IDENTITY(1,1) PRIMARY KEY,
    ApplicationID INT,
    PersonID INT,
    RGID INT,
    PropertyID INT,
    PreviousApplicationStatus VARCHAR(20),
    ApplicationDate DATE,
    WithdrawalDate DATETIME DEFAULT GETDATE()
)
END
GO

CREATE TRIGGER WithdrawnApplications
ON ApplicationSubmission
AFTER DELETE
AS
BEGIN
    INSERT INTO ApplicationWithdrawals (ApplicationID, PersonID, RGID, PropertyID, PreviousApplicationStatus, ApplicationDate, WithdrawalDate)
    SELECT ApplicationID, PersonID, RGID, PropertyID, ApplicationStatus, ApplicationDate, GETDATE()
    FROM deleted
END
GO
```

Data Encryption

```
-- Create a Database Master Key
IF NOT EXISTS (
    SELECT 1
    FROM sys.symmetric_keys
    WHERE name = '##MS_DatabaseMasterKey##'
)
BEGIN
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MasterKey';
END

-- Verify that master key exists
SELECT name KeyName, symmetric_key_id KeyID, key_length KeyLength, algorithm_desc KeyAlgorithm
FROM sys.symmetric_keys
```

```
-- Column Level Encryption for SSN of Property Managers

-- Create a Certificate
IF NOT EXISTS (
    SELECT 1
    FROM sys.certificates
    WHERE name = 'PMSSN'
)
BEGIN
CREATE CERTIFICATE PMSSN
WITH SUBJECT = 'Property Manager SSN';
END


-- Create a Symmetric Key and associate it with the Certificate
IF NOT EXISTS (
    SELECT 1
    FROM sys.symmetric_keys
    WHERE name = 'PMSSNKey'
)
BEGIN
CREATE SYMMETRIC KEY PMSSNKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE PMSSN;
END
```

```
-- Open the Symmetric Key
OPEN SYMMETRIC KEY PMSSNKey
DECRYPTION BY CERTIFICATE PMSSN;

-- Update data with encryption
UPDATE PropertyManager SET SSN = ENCRYPTBYKEY(KEY_GUID('PMSSNKey'), SSN);

-- Close the Symmetric Key
CLOSE SYMMETRIC KEY PMSSNKey;
GO

-- Check if data is encrypted
SELECT * FROM PropertyManager;


-- Open the Symmetric Key
OPEN SYMMETRIC KEY PMSSNKey
DECRYPTION BY CERTIFICATE PMSSN;

-- Retrieve data with decryption
SELECT OwnerID, NoOfPropertiesManaged,
       CAST(DECRYPTBYKEY(SSN) AS VARCHAR(16)) AS DecryptedSSN
FROM PropertyManager;

-- Close the Symmetric Key
CLOSE SYMMETRIC KEY PMSSNKey;
```

Non Clustered Index

A non-clustered index is a database structure that improves search speed by creating a separate, logically ordered list of key values with pointers to the actual data rows. It doesn’t alter the physical order of the table’s data and is useful for queries involving **WHERE**, **JOIN**, or **ORDER BY** operations.

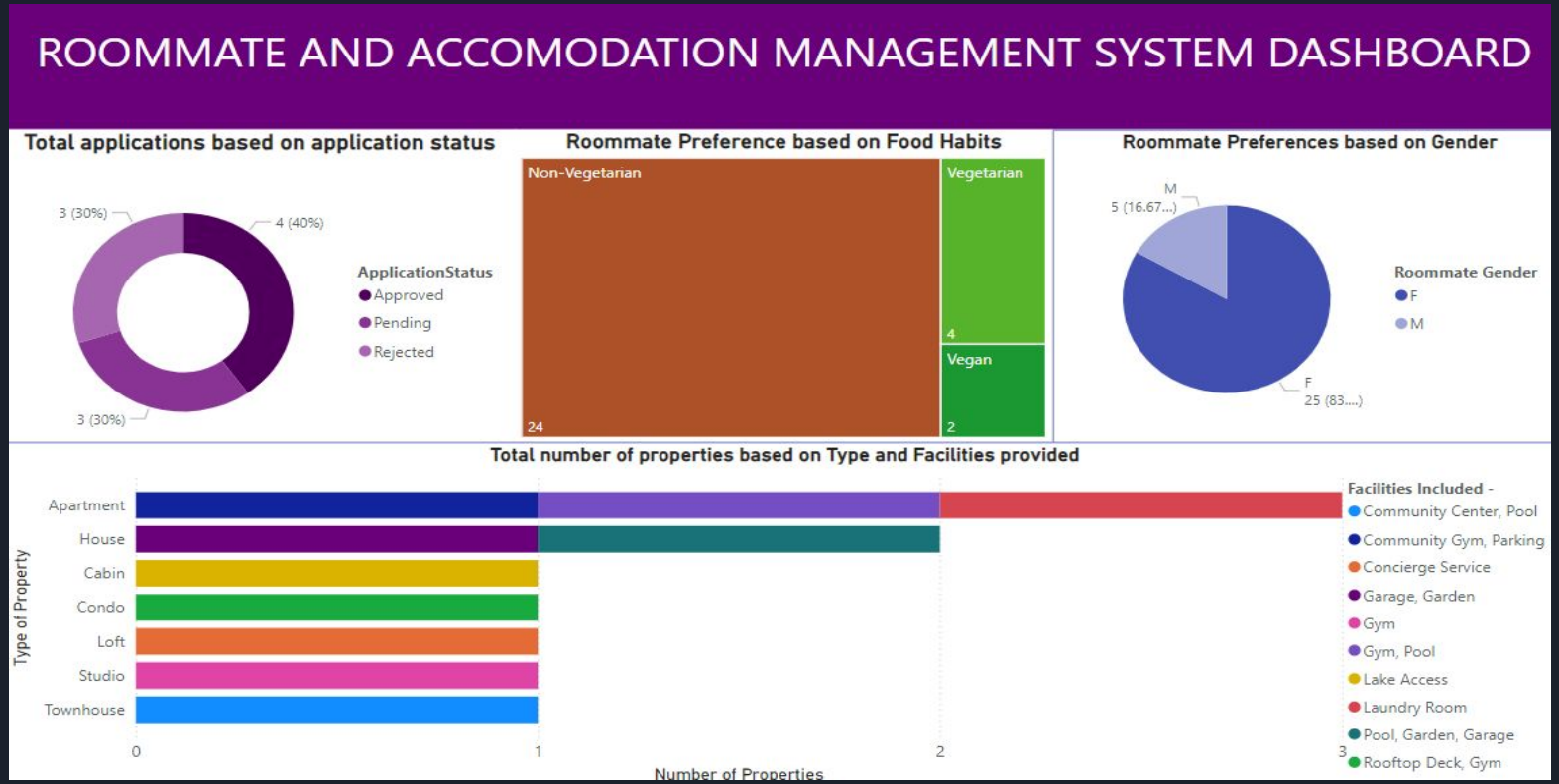
Non Clustered Index

```sql
-- This index improves search performance for filtering
-- roommates by gender and language.
IF NOT EXISTS (
    SELECT 1
    FROM sys.indexes
    WHERE name = 'IX_Roommates_Gender_Languages'
        AND object_id = OBJECT_ID('Roommates')
)
BEGIN
    CREATE NONCLUSTERED INDEX IX_Roommates_Gender_Languages
    ON Roommates (Gender, Languages);
END;
```

```sql
-- This index enhances performance for queries retrieving
-- roommate groups based on their RentSplit.
IF NOT EXISTS (
    SELECT 1
    FROM sys.indexes
    WHERE name = 'IX_RoommatesGroup_RentSplit'
        AND object_id = OBJECT_ID('RoommatesGroup')
)
BEGIN
    CREATE NONCLUSTERED INDEX IX_RoommatesGroup_RentSplit
    ON RoommatesGroup (RentSplit);
END;
```

```sql
-- This index is designed to improve performance for queries filtering by ApartmentType, RentRange, and RoommateGender.
IF NOT EXISTS (
    SELECT 1
    FROM sys.indexes
    WHERE name = 'IX_Preferences_ApartmentType_RentRange_RoommateGender'
        AND object_id = OBJECT_ID('Preferences')
)
BEGIN
    CREATE NONCLUSTERED INDEX IX_Preferences_ApartmentType_RentRange_RoommateGender
    ON Preferences (ApartmentType, RentRange, RoommateGender);
END;
```

Power BI Report



LIVE DEMO

Conclusion

- In conclusion, ROOM & MATES is more than a web application—it is a transformative solution designed to make finding the perfect accommodation and compatible roommates seamless, personalized, and stress-free.

- By addressing the challenges faced by individuals, particularly those relocating to new regions, it fosters a sense of security, convenience, and community.

- With its user-centric design, innovative features, and focus on enhancing the housing search experience, ROOM & MATES empowers users to build not just a living space, but a home. It’s the ideal companion for modern living in an increasingly connected world.

Thank You