NORMALIZATION DOCUMENT – Housing Management System (HMS)

Group 3:

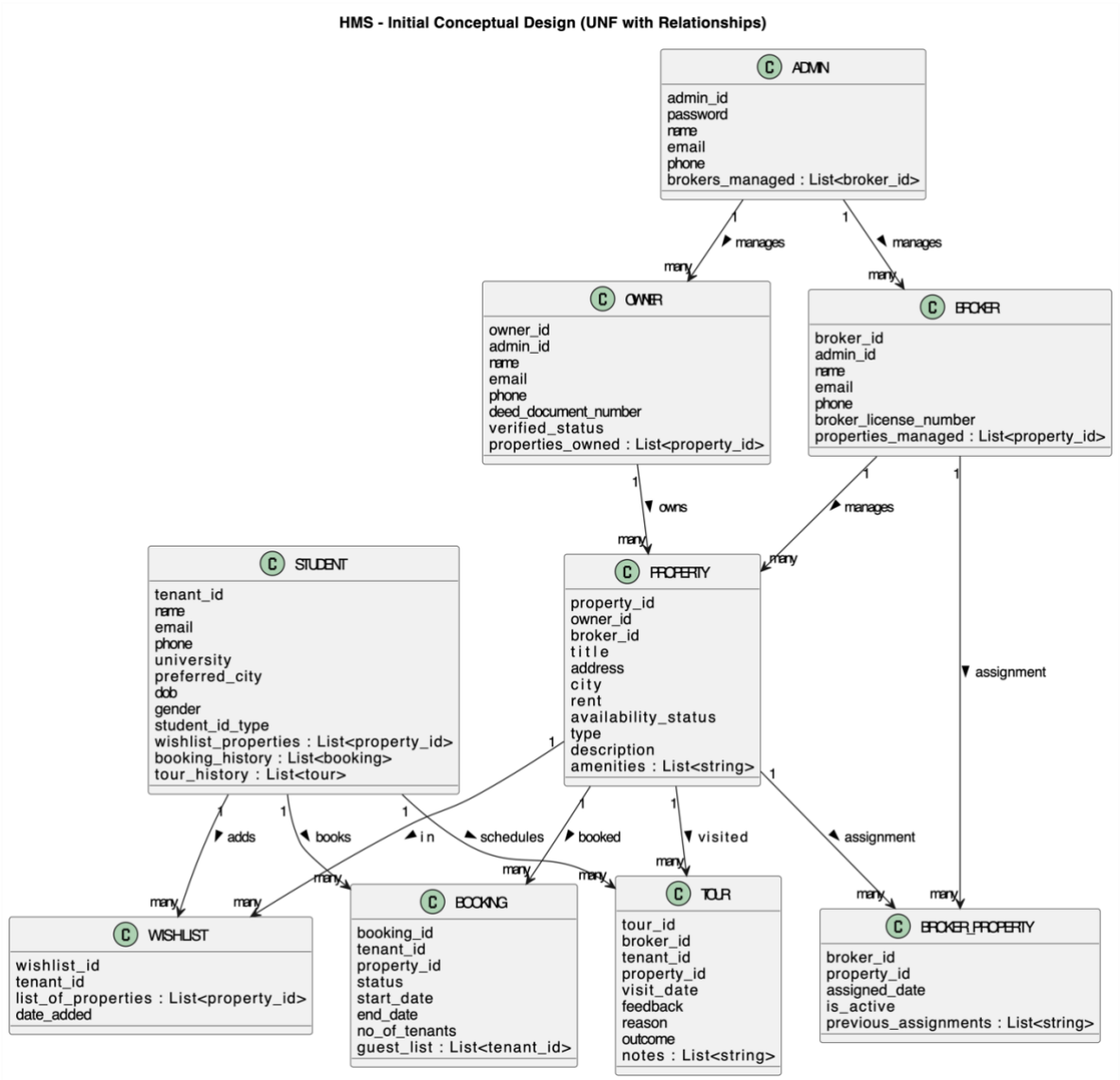
Tanya Bansal- 002020208

Prasham Nagda-002539021

Neerajaa Kadam- 002509421

1.1 INITIAL CONCEPTUAL DESIGN OF HMS (UNF)

Unnormalized Conceptual Model — BEFORE 1NF, 2NF, 3NF. This is our **raw, original conceptual design** before any normalization rules were applied.



2.1 Why Normalization Was Necessary

In the initial conceptual design of HMS, entities like **ADMIN, OWNER, BROKER, STUDENT, PROPERTY, WISHLIST, BOOKING, TOUR, BROKER_PROPERTY** had:

- **Repeating groups / lists**
  - brokers_managed for an admin
  - properties_managed for a broker
  - properties_owned for an owner
  - wishlist_properties, booking_history, tour_history for a student
  - list_of_properties in wishlist
  - guest_list in booking
  - notes in tour
  - previous_assignments in broker_property
- **Multi-valued attributes**
  - amenities in property as a list (WiFi, Laundry, Parking, etc.)
- **Composite-key based dependencies**
  - (broker_id, property_id) in BROKER_PROPERTY

If we kept this design, we'd get:

- **Update anomalies** – Same broker/owner/student data repeated in multiple rows.
- **Insert anomalies** – Can't insert a student without a wishlist; can't add a property without all related lists.
- **Delete anomalies** – Deleting a booking might remove the only copy of some relationship.
- **Redundancy and inconsistency** – Phone/email repeated many times.

Normalization was required to:

- Make each table store **one type of fact**
- Ensure every non-key attribute depends **only** on the primary key
- Remove repeating groups & multi-valued fields
- Reduce redundancy and anomalies

2.2 Functional Dependencies at UNF Stage

Entity	Functional Dependencies
ADMIN	admin_id → password, name, email, phone, brokers_managed
OWNER	owner_id → admin_id, name, email, phone, deed_document_number, verified_status, properties_owned
BROKER	broker_id → admin_id, name, email, phone, broker_license_number, properties_managed

Entity	Functional Dependencies
PROPERTY	property_id → owner_id, broker_id, title, address, city, rent, availability_status, type, description, amenities
STUDENT	tenant_id → name, email, phone, university, preferred_city, dob, gender, student_id_type, wishlist_properties, booking_history, tour_history
WISHLIST	wishlist_id → tenant_id, list_of_properties, date_added
BOOKING	booking_id → tenant_id, property_id, status, start_date, end_date, no_of_tenants, guest_list
TOUR	tour_id → broker_id, tenant_id, property_id, visit_date, feedback, reason, outcome, notes
BROKER_PROPERTY	(broker_id, property_id) → assigned_date, is_active, previous_assignments

At the UNF stage, each entity exhibits dependencies based on natural identifiers. Multi-valued and composite determinants appear in the UNF model and are eliminated progressively in 1NF, 2NF, and 3NF.

---

3. First Normal Form (1NF)

3.1 Why 1NF Was Necessary

1NF requires:

- No repeating groups
- No multi-valued attributes
- All fields are **atomic**

We had lists and repeating groups everywhere, so we:

- Split lists into **separate rows** (e.g., WISHLIST, BOOKING)
- Kept only atomic attributes in each row
- Moved many-to-many relationships into **separate linking tables** (e.g., BROKER_PROPERTY)

3.2 BEFORE vs AFTER (UNF → 1NF) – Key Tables

ADMIN (1NF Before -> After)

admin_id	INT
password	VARCHAR2
name	VARCHAR2
email	VARCHAR2

phone	VARCHAR2
brokers_managed(list)	LIST<INT>

admin_id	INT (PK)
password	VARCHAR2 (255)
name	VARCHAR2 (100)
email	VARCHAR2 (100)
phone	VARCHAR2 (15)

-- brokers managed is represented via BROKER.admin_id

OWNER (1NF Before -> After)

owner_id	INT
admin_id	INT
name	VARCHAR2
email	VARCHAR2
phone	VARCHAR2
deed_document_number	INT
verified_status	VARCHAR2
properties_owned	LIST<INT>

owner_id	INT (PK)
admin_id	INT (FK → ADMIN.admin_id)
name	VARCHAR2 (100)
email	VARCHAR2 (100)
phone	VARCHAR2 (15)
deed_document_number	INT
verified_status	VARCHAR2 (20)

-- properties owned are represented by PROPERTY.owner_id

BROKER (1NF Before -> After)

broker_id	INT
admin_id	INT
name	VARCHAR2
email	VARCHAR2
phone	VARCHAR2
broker_license_number	INT
properties_managed	LIST<INT>

broker_id	INT (PK)
admin_id	INT (FK)
name	VARCHAR2 (100)
email	VARCHAR2 (100)
phone	VARCHAR2 (15)
broker_license_number	INT

-- managed properties handled via BROKER_PROPERTY

PROPERTY (1NF Before -> After)

property_id	INT
owner_id	INT
broker_id	INT
title	VARCHAR2
address	VARCHAR2
city	VARCHAR2
rent	DECIMAL
availability_status	VARCHAR2
type	VARCHAR2
description	VARCHAR2
amenities	LIST<VARCHAR2>

property_id	INT (PK)
owner_id	INT (FK)
broker_id	INT (FK)
title	VARCHAR2 (100)
address	VARCHAR2 (200)
city	VARCHAR2 (100)
rent	DECIMAL (10,2)
availability_status	VARCHAR2 (20)
type	VARCHAR2 (50)
description	VARCHAR2 (255)
amenities	VARCHAR2 (255)

amenities -- stored as atomic text (comma-separated string)

STUDENT (1NF Before -> After)

tenant_id	INT
name	VARCHAR2
email	VARCHAR2
phone	VARCHAR2

university	VARCHAR2
preferred_city	VARCHAR2
dob	DATE
gender	VARCHAR2
student_id_type	VARCHAR2
wishlist_properties	LIST<INT>
booking_history	LIST<INT>
tour_history	LIST<INT>

tenant_id	INT (PK)
name	VARCHAR2(100)
email	VARCHAR2(100)
phone	VARCHAR2(15)
university	VARCHAR2(100)
preferred_city	VARCHAR2(100)
dob	DATE
gender	VARCHAR2 (10)
student_id_type	VARCHAR2 (15)

-- wishlist, bookings, tours handled in separate tables

WISHLIST (1NF Before -> After)

wishlist_id	INT
tenant_id	INT
list_of_properties	LIST<INT>
date_added	DATE

wishlist_id	INT (PK)
tenant_id	INT (FK)
property_id	INT (FK)
date_added	DATE

-- one property per row

BOOKING (1NF Before -> After)

booking_id	INT
tenant_id	INT
property_id	INT

status	VARCHAR2
start_date	DATE
end_date	DATE
no_of_tenants	INT
guest_list	LIST <INT>

booking_id	INT (PK)
tenant_id	INT (FK)
property_id	INT (FK)
status	VARCHAR2 (20)
start_date	DATE
end_date	DATE
no_of_tenants	INT

-- guest_list list removed; captured via no_of_tenants

TOUR (1NF Before -> After)

tour_id	INT
broker_id	INT
tenant_id	INT
property_id	INT
visit_date	DATE
feedback	VARCHAR2
reason	VARCHAR2
outcome	VARCHAR2
notes	LIST<VARCHAR2>

tour_id	INT (PK)
broker_id	INT (FK)
tenant_id	INT (FK)
property_id	INT (FK)
visit_date	DATE
feedback	VARCHAR2 (50)
reason	VARCHAR2 (1000)
outcome	VARCHAR2 (255)

-- notes list removed or stored outside this core schema

BROKER_PROPERTY (1NF Before -> After)

broker_id	INT
property_id	INT
assigned_date	DATE
is_active	BOOLEAN
previous_assignments	LIST <DATE>

broker_id	INT (Composite PK)
property_id	INT (Composite PK)
assigned_date	DATE
is_active	CHAR (1)

-- each assignment as one row; no list of previous_assignments

After 1NF, all columns are **atomic**, and repeating groups are removed.

---

4. Second Normal Form (2NF)

4.1 Why 2NF Was Necessary

2NF applies **only** to tables that have a **composite primary key**.
A table violates 2NF when a **non-key attribute depends on only part of the composite key** (partial dependency).

In the HMS system, **BROKER_PROPERTY** was the only table that used a composite primary key: (broker_id, property_id). No other HMS table had a composite primary key; therefore, they already satisfied 2NF and did not require decomposition.

Although all non-key attributes in BROKER_PROPERTY already depended on the full composite key (no actual partial dependency existed), using a composite primary key can still lead to:

- complex joins
- difficulty referencing the table from other tables
- repeated composite-key usage
- poor indexing performance

Therefore, the design replaces the composite key with a single surrogate key.

Solution:
Introduce a **surrogate primary key (**broker_property_id**)**, ensuring all attributes depend fully on a single PK.

4.2 BEFORE vs AFTER (1NF → 2NF)

BROKER_PROPERTY

broker_id	INT (PK)
property_id	INT (PK)
assigned_date	DATE
is_active	CHAR(1)

FDs:
(broker_id, property_id) → assigned_date, is_active

No partial dependency
(both attributes depend on BOTH parts)

After 2NF:

Broker_property_id	INT (PK)
broker_id	INT (FK)
property_id	INT (FK)
assigned_date	DATE
Is_active	CHAR(1)

FD:
broker_property_id → broker_id, property_id, assigned_date, is_active

Although the BROKER_PROPERTY table did not contain a partial dependency, it used a composite primary key, which is less efficient for relational design. To improve usability and maintain consistency with the rest of the schema, a surrogate primary key (broker_property_id) was introduced, making the table fully compliant with 2NF.

---

5. Third Normal Form (3NF)

5.1 Why 3NF Was Necessary

3NF removes **transitive dependencies**:
A non-key attribute should **not depend on another non-key attribute**.

Example pattern that 3NF avoids:

- property_id → owner_id → owner_phone
  (so property_id indirectly determines owner_phone)

How our design avoids this:

Instead of storing owner/broker/student details inside other tables:

- **OWNER** stores owner attributes
- **BROKER** stores broker attributes
- **STUDENT** stores student attributes
- **PROPERTY**, **BOOKING**, **WISHLIST**, **TOUR**, and **BROKER_PROPERTY** contain **only foreign keys**, not the actual details

This ensures:

- No non-key attribute depends on another non-key attribute
- Every attribute depends *only* on its table's primary key

Thus, the HMS schema **achieves 3NF**.

---

6. Final 3NF Schema

These are implemented tables, and they are in **3NF**:

Table	**Attributes**	**Functional Dependency**	**3NF Status & Reason**
ADMIN	admin_id (PK), password, name, email, phone	admin_id → password, name, email, phone	✓ In 3NF — all non-key attributes depend only on PK; no transitive dependencies
OWNER	owner_id (PK), admin_id (FK), name, email, phone, deed_document_number, verified_status	owner_id → admin_id, name, email, phone, deed_document_number, verified_status	✓ In 3NF — no non-key attribute determines another non-key attribute
BROKER	broker_id (PK), admin_id (FK), name, email, phone, broker_license_number	broker_id → admin_id, name, email, phone, broker_license_number	✓ In 3NF — all attributes depend directly on PK
STUDENT	tenant_id (PK), name, email, phone, university, preferred_city, dob, gender, student_id_type	tenant_id → all attributes	✓ In 3NF — no transitive dependencies
PROPERTY	property_id (PK), owner_id (FK), broker_id (FK), title, address, city, rent,	property_id → owner_id, broker_id, title, address, city, rent,	✓ In 3NF — owner/broker details stored separately →

Table	**Attributes**	**Functional Dependency**	**3NF Status & Reason**
	availability_status, type, description, amenities	availability_status, type, description, amenities	no transitive dependency
WISHLIST	wishlist_id (PK), tenant_id (FK), property_id (FK), date_added	wishlist_id → tenant_id, property_id, date_added	✓ In 3NF — all attributes directly depend on PK; no non-key → non-key dependency
BOOKING	booking_id (PK), tenant_id (FK), property_id (FK), status, start_date, end_date, no_of_tenants, created_at	booking_id → tenant_id, property_id, status, start_date, end_date, no_of_tenants, created_at	✓ In 3NF — tenant/property details separated; no transitive dependency
TOUR	tour_id (PK), broker_id (FK), tenant_id (FK), property_id (FK), visit_date, feedback, reason, outcome	tour_id → broker_id, tenant_id, property_id, visit_date, feedback, reason, outcome	✓ In 3NF — no attribute depends on another non-key attribute
BROKER_PROPERTY	broker_property_id (PK), broker_id (FK), property_id (FK), assigned_date, is_active	broker_property_id → broker_id, property_id, assigned_date, is_active	✓ In 3NF — surrogate PK removes all composite-key issues; no transitive dependency

Housing Management System - Entity Relationship Diagram