Section I: Cover Page

Real Estate Property Management System
Shez Rahman
Student ID: 916867424
Github username: shezgo

Version History Table:

Milestone	**Date submitted**
M1V1	9/17/2024
M2V1	10/03/2024
M1V2	10/03/2024
M2V2	10/19/2024
M3V1	10/19/2024

Section II: Table of Contents

Section III: Project Description

The motivation for creating this database system for real estate property management is having worked with many investors during my time as a realtor, there are many factors to consider when acquiring a property. This includes maintenance, the existing tenants, future tenants, and any equity plays in progress. Aspiring investors may often lose sight of other variables by zooming in on one, such as cash flow, which can lead to a loss in factoring in equity plays or varying initial investments due to required rehabilitation. This database system aims to enhance the investor's view and reduce the struggle to strategize by consolidating and storing info on these different variables in a singular database for investors and property managers to keep track of their investments.

The database system will function by storing data on each property's current status regarding tenants and needed maintenance. In addition, a unique feature of this database is to couple this with additional data: for investors who are interested in home-flipping or refinancing strategies to grow their portfolios, equity plays will be encouraged by storing entry purchase prices and target after-rehab values. This works to account for projects in progress - relevant values, IE rehab costs, will also be included in the database. Because strong cash-flow continues to become scarcer in high ticket areas such as the Bay Area or Seattle, it's important to create a shift in how investments are perceived - this organization of data allows investors to factor in both rental income and equity.

ManageCasa and Buildium are two property management software tools that would benefit from this because they focus more on simple accounting, tenant management, and maintenance management, but do not track ongoing projects. They would benefit from using this database to provide investors with the tools needed to support equity oriented strategies as well as manage ongoing accounting and maintenance.

1. **Use case:** Decide whether to sell or keep a property (using past expenditures and expected ARV)
   Actor: Owner of properties (John)
   Description: John owns 3 properties, and has been having trouble with one of the tenants for the last year since he purchased his most recent property. The tenant is constantly texting him all day and night with various requests for maintenance. The stress has been making him consider if he should keep the property. By using the Real Estate Property Management Database System, he looks up the records of the property and finds that the tenant has always paid their rent on time. In addition, the property is his top performer in terms of rental income cash flow. As he's pulling data on the property, he notices the after repair value that he targeted when he first purchased the

property, which was a whopping 30% higher than the entry level. As a result, he decides to hire a property manager to eliminate direct contact with the tenant, and pour more money into the rehab of the property to raise its value to the target after-rehab value.

2. **Use case:** Evicting a tenant
   Actor: Owner of properties (Alex), Property manager (Jess), Tenant (Sarah)
   Description: Jess is managing a number of properties for Alex, and has been struggling with a tenant, Sarah, who refuses to stop smoking inside the property. Sarah also makes excessive noise during quiet hours. Because of her habits, tenants keep terminating their leases because sharing a wall with Sarah is unpleasant due to the smell and noise. Jess needs to build a case to convince Alex that an eviction is worth the effort here, because Alex doesn't know how troublesome this tenant has been. Using the Real Estate Property Management Database System, Jess is able to pull Sarah's lease which explicitly states that smoking in the unit is grounds for termination of the lease, as well as a consistent record of delinquent rent payments. In addition, Jess also pulls a series of notes of every time Sarah has been given notice to cease smoking indoors. Lastly, because Sarah has been in the unit for 7 years, the unit is very outdated, so if it's vacant, the value of the property can be increased via rehab for a higher refinance amount. Armed with a starter pack of evidence to support Sarah's eviction, Jess sends the data to Alex to persuade him that it's worth the investment for a stable and lucrative property.

3. **Use case:** Accounting to determine when next property can be purchased
   Actor: Owner of properties (Liz), Accountant (Eric)
   Description: Liz is eager to grow her portfolio from its current state of 3 properties and 10 units, but has no liquidity. She does not want to sell any properties, and is not savvy when it comes to different ways of purchasing a property without having cash in hand. She contacts Eric, her accountant, and asks him when she'll be able to buy her next property. Eric pulls the net cash flow on each property and totals them to get a total number of $5k income per month. For her target purchase price of $1m, she would need to wait 40 months for the cash flow to cover a 20% down payment, not accounting for closing costs. Eric finds that she hadn't refinanced on any of her properties for the last 5 years and is likely sitting on a substantial amount of equity. He recommends to her that she refinance on one of her properties in order to enter the market within a couple of months, rather than wait 3-4 years for her next acquisition.

Section IV: Functional Database Requirements

1. Users
   1.1 A user shall create at most one account
   1.2 A user shall have a unique tracking id
   1.3 A user, without an account, shall not be able to view a portfolio

2. RegisteredUsers
   2.1 A registered user is also a user
   2.2 A registered user shall have a password
   2.3 A registered user shall have a portfolio
   2.4 A registered user shall have at most one unique email
   2.5 A registered user shall have a first name
   2.6 A registered user shall have a last name
   2.7 A registered user shall have a full name
   2.8 A registered user shall have a tracking id

3. Portfolios
   3.1 A portfolio shall have many properties
   3.4 A portfolio shall have a unique id
   3.5 A portfolio shall have a number of properties
   3.6 A portfolio shall have a last appraised value
   3.7 A portfolio shall belong to at least one registered user

4. Properties
   4.1 A property shall have at least one address
   4.2 A property shall have many tenants
   4.3 A property shall have a quantity of units
   4.4 A property shall have a total rental income
   4.5 A property shall have a monthly capital expenditures amount
   4.6 A property shall have a property history
   4.7 A property shall have a bedroom count
   4.8 A property shall have a bathroom count
   4.9 A property shall have a square footage size
   4.10 A property shall have a lot size
   4.11 A property shall have a target after rehab value

4.12 A property shall have a project info
4.13 A property shall have units
4.14 A property shall have a property type
4.15 A property shall belong to at least one portfolio
4.16 A property shall have a unique id

5. Addresses
   5.1 An address shall have one number
   5.2 An address shall have one street name
   5.3 An address shall have a city
   5.4 An address shall have a state/province
   5.5 An address shall have a country
   5.7 An address shall have a unique id
   5.8 An address shall have a numbered street

6. Tenants
   6.1 A tenant shall have a first name
   6.2 A tenant shall have a last name
   6.3 A tenant shall have a full name
   6.4 A tenant shall have a lease agreement
   6.7 A tenant shall have a payment history
   6.9 A tenant shall belong to a unit
   6.10 A tenant shall have a notes field
   6.11 A tenant shall have a unique id

7. PropertyHistories
   7.1 A propertyhistory shall have a purchase date
   7.2 A propertyhistory shall have a purchase amount
   7.3 A propertyhistory shall have a maintenance notes text field
   7.4 A propertyhistory shall have a last appraised value
   7.6 A propertyhistory shall have many expense histories
   7.7 A propertyhistory shall have inspection records
   7.8 A propertyhistory shall have a unique id

8. Units
   8.1 A unit shall have a bedroom count
   8.2 A unit shall have a bathroom count
   8.3 A unit shall have a rent
   8.4 A unit shall have many tenants
   8.5 A unit shall be vacant or occupied
   8.6 A unit shall belong to a property
   8.7 A unit shall have a payment history
   8.8 A unit shall have a unique id

8.9 A unit shall have a building number

9. PaymentHistories
    9.1 A paymenthistory shall belong to a tenant
    9.2 A paymenthistory shall belong to a unit
    9.3 A paymenthistory shall have a unique id
    9.5 A paymenthistory shall have an amount
    9.6 A paymenthistory shall have a paid date
    9.9 A paymenthistory shall have a due date

10. ExpenseHistories
    10.1 An expensehistory shall have a label
    10.2 An expensehistory shall have a cost
    10.3 An expensehistory shall have a date
    10.4 An expensehistory shall belong to a propertyhistory
    10.6 An expensehistory shall have a unique id

11. ProjectInfos
    11.1 A projectinfo shall belong to a property
    11.2 A projectinfo shall have a project title
    11.3 A projectinfo shall have a project description
    11.4 A projectinfo shall have an in progress flag
    11.6 A projectinfo shall have a unique id
    11.7 A projectinfo shall have many projectupdates
    11.8 A projectinfo shall have zero or many contractors

12. InsurancePolicies
    12.1 An insurancepolicy shall belong to a property
    12.2 An insurancepolicy shall have a policy number
    12.3 An insurancepolicy shall have a provider
    12.4 An insurancepolicy shall have a monthly cost
    12.5 An insurancepolicy shall have a start date
    12.6 An insurancepolicy shall have an end date
    12.7 An insurancepolicy shall have a unique id

13. Contractors
    13.1 A contractor shall have a unique id
    13.2 A contractor shall have a company name
    13.3 A contractor shall have a first name
    13.4 A contractor shall have a last name
    13.5 A contractor shall have a full name
    13.6 A contractor shall have a services field

14. InspectionRecords
    14.1 An inspectionrecord shall belong to a propertyhistory
    14.2 An inspectionrecord shall have a unique id
    14.3 An inspectionrecord shall have an inspector name
    14.4 An inspectionrecord shall have a notes

15. LeaseAgreements
    15.1 A leaseagreement shall belong to many tenants
    15.2 A leaseagreement shall have a unique id
    15.3 A leaseagreement shall have a start date
    15.4 A leaseagreement shall have an end date
    15.5 A leaseagreement shall have many tenants
    15.6 A leaseagreement shall have a terms field
    15.7 A leaseagreement shall have a property id
    15.8 A leaseagreement shall have a rent

16. TenantLeaseAgreements (Associative)
    16.1 A tenantleaseagreement shall have a tenant
    16.2 A tenantleaseagreement shall have a leaseagreement
    16.3 A tenantleaseagreement shall have a unique id tracker
    16.4 A tenantleaseagreement shall have a tenant name

17. TaxRecords
    17.1 A taxrecord shall have a unique id
    17.2 A taxrecord shall have a year
    17.3 A taxrecord shall have an amount due
    17.4 A taxrecord shall have an amount paid
    17.5 A taxrecord shall have a due date
    17.6 A taxrecord shall have a payment date
    17.7 A taxrecord shall have a property id
    17.8 A taxrecord shall belong to a property

18. Mortgages
    18.1 A mortgage shall have a unique id
    18.2 A mortgage shall have a lender name
    18.3 A mortgage shall have a loan principal balance
    18.4 A mortgage shall have an interest rate
    18.5 A mortgage shall have a total monthly payment
    18.6 A mortgage shall have a start date
    18.7 A mortgage shall have an end date
    18.8 A mortgage shall have a property id
    18.9 A mortgage shall have a terms

19. UserPortfolios

19.1 A UserPortfolio shall have a RegisteredUser
19.2 A UserPortfolio shall have a Portfolio
19.3 A UserPortfolio shall have a tracking id
19.4 A UserPortfolio shall have a last appraised value

20. ProjectUpdates
    21.1 A projectupdate shall belong to a projectinfo
    21.2 A projectupdate shall have a date
    21.3 A projectupdate shall have an update text field
    21.4 A projectupdate shall have a unique tracking ID

21. ProjectContractors
    22.1 A projectcontractor shall belong to a projectinfo
    22.2 A projectcontractor shall have a contractor
    22.3 A projectcontractor shall have a tracking id
    22.4 A projectcontractor shall have a services field

22. Roles
    23.1 A role for a registered user shall be an admin
    23.2 A role for a registered user shall be an owner
    23.3 A role shall have a role id

23. ProjectInfoUpdates
    24.1 A projectinfoupdate shall have a projectinfo
    24.2 A projectinfoupdate shall have a projectupdate
    24.3 A projectinfoupdate shall have a date
    24.4 A projectinfoupdate shall have a project title

24. UnitTenants
    25.1 A unittenant shall have a unit
    25.2 A unittenant shall have a tenant
    25.3 A unittenant shall reference a numbered street
    25.4 A unittenant shall have a tenant name

25. PortfolioProperties
    26.1 A portfolioproperty shall have a portfolio
    26.2 A portfolioproperty shall have a property
    26.3 A portfolioproperty shall have a property's rent

26. UnitPaymentHistories
    27.1 A unitpaymenthistory shall have a unit
    27.2 A unitpaymenthistory shall have a paymenthistory
    27.3 A unitpaymenthistory shall have a paid date

27.4 A unitpaymenthistory shall have a building number
27.5 A unitpaymenthistory shall have an amount

27. TenantPaymentHistories
    28.1 A tenantpaymenthistory shall have a tenant
    27.2 A tenantpaymenthistory shall have a paymenthistory
    27.3 A tenantpaymenthistory shall have a paid date
    27.4 A unitpaymenthistory shall have a tenant name
    27.5 A unitpaymenthistory shall have an amount

Section V: Non-functional Database Requirements

1. Performance
   1.1 The database shall return data in under 3 seconds for 95% of the queries
   1.2 The database shall use indexes to minimize query execution time and prevent degradation as data size increases
   1.3 The system shall process bulk data of 1000 records or more in under 5 minutes
2. Security
   2.1 All sensitive data, including passwords and financial info, shall be encrypted with AES encryption
   2.2 Access to the database shall be restricted based on user privileges
   2.3 The system shall implement measures such as input validation to prevent SQL injection
3. Scalability
   3.1 The database system shall support the addition of new database instances by load balancing to accommodate increased data volume without disrupting service as needed
   3.2 The database system shall incorporate load balancing to distribute query traffic across servers as needed
   3.3 The database system shall support at least 1000 concurrent connections
4. Capability
   4.1 The database system shall support real-time synchronization so that changes are instantly reflected to all users
   4.2 The database system shall manage concurrent data access and updates correctly by using mechanisms like locking to avoid race conditions
   4.3 Users shall be able to export and import data in CSV, XML, and JSON formats
5. Environmental
   5.1 The database system shall be optimized to minimize power consumption
   5.2 The database system shall support methods of recovery in case of environmental disaster
   5.3 The database system shall use data centers and technology that are environmentally regulated
6. Coding Standard
   6.1 The database system shall use camelCase for all fields and plural names for all tables

6.2 All code shall have supportive documentation, such as a README, or comments

6.3 All database changes shall be managed in a version control system

7. Storage

   7.1 The database shall assign 10 MB of memory per tables

   7.2 The database system shall support persistent storage

   7.3 The database system shall be able to be backed up in case of emergencies

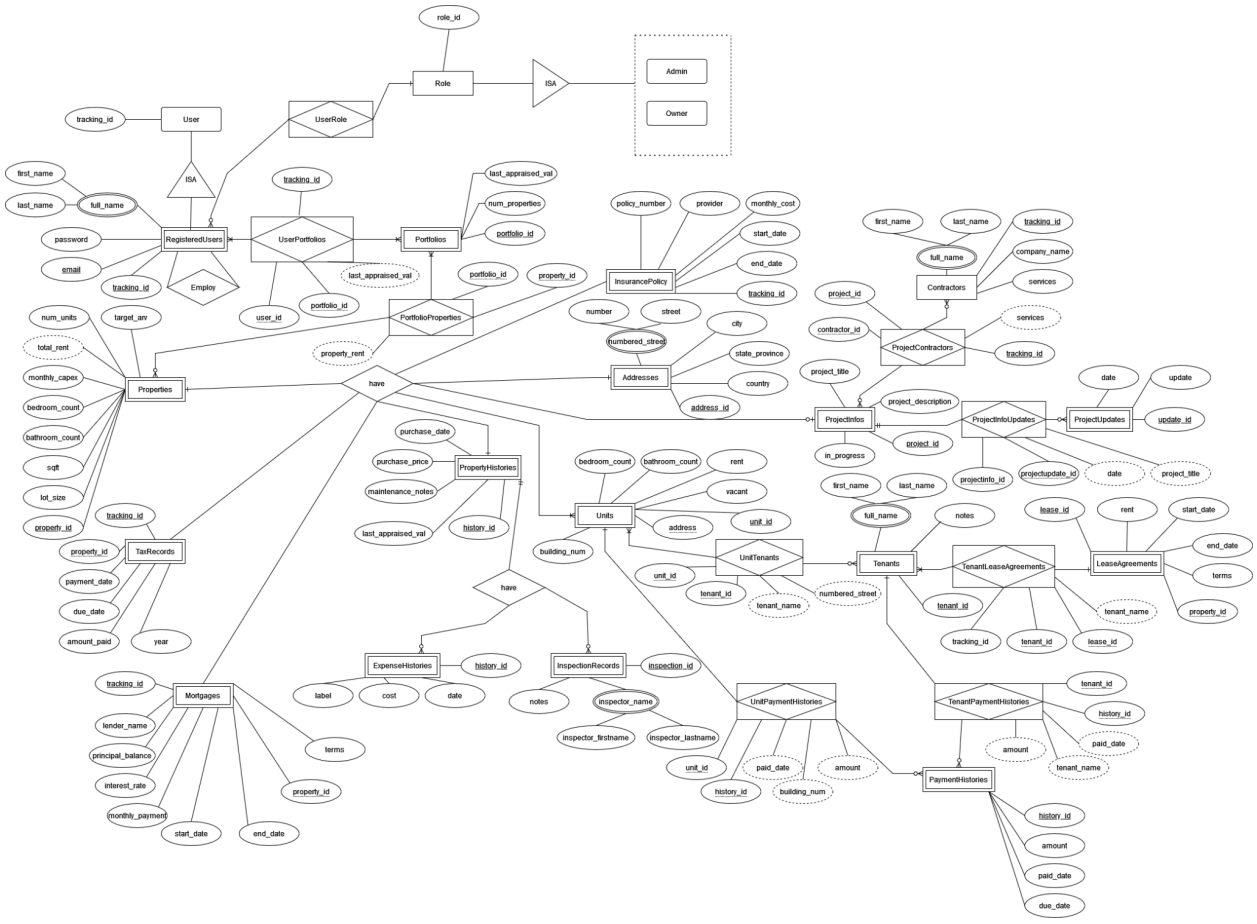
8. Privacy

   8.1 Sensitive information such as passwords and financial info shall be protected from being displayed to users who don't have sufficient privileges

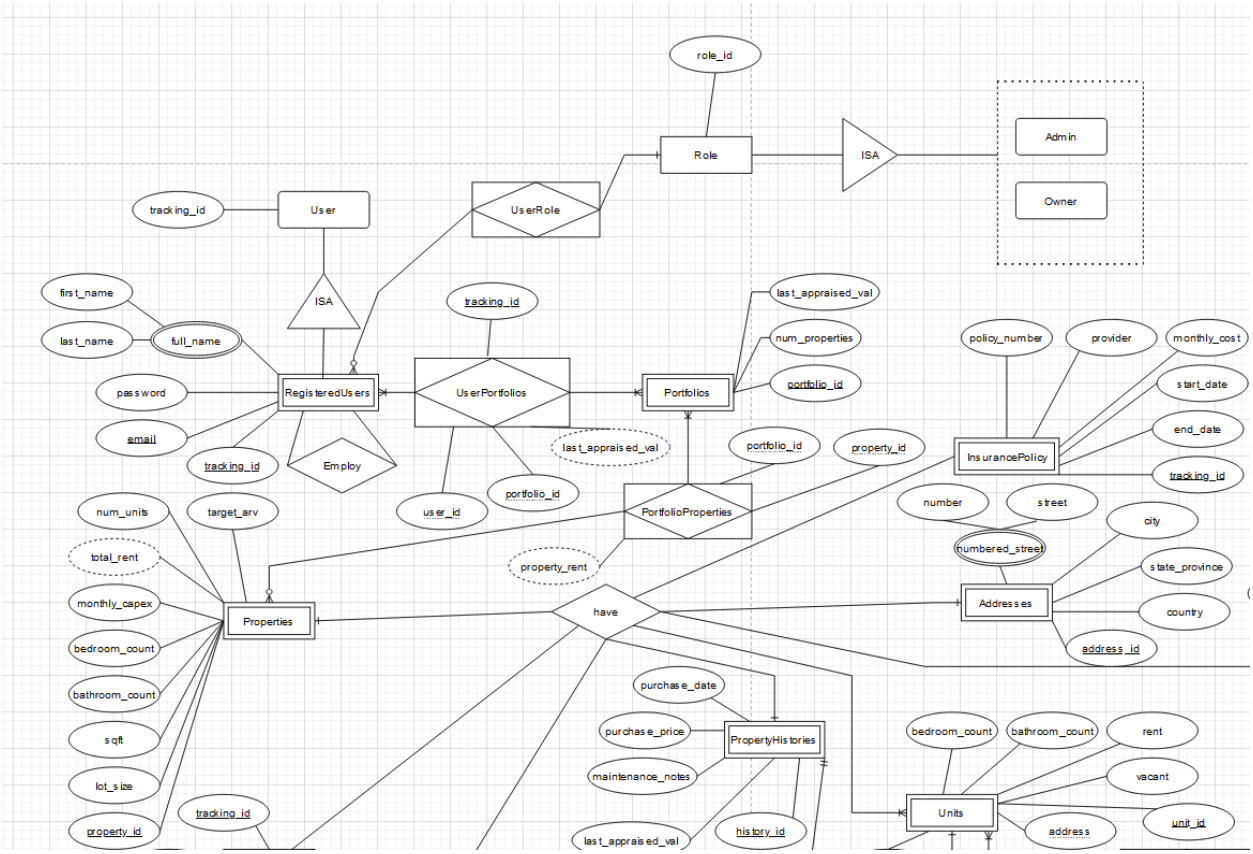
   8.2 The system shall log whenever personal data is accessed or modified and by whom

   8.3 Tenant information shall only be viewable by users with permission

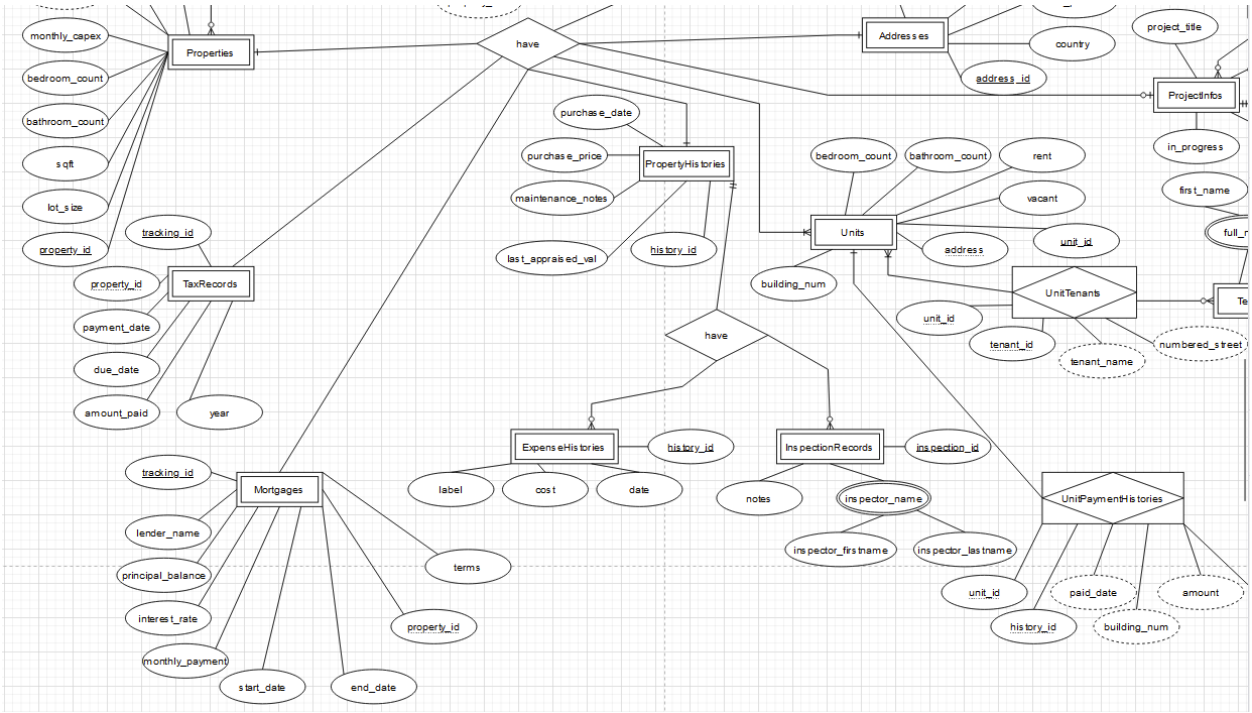
Section VI: Entity Relationship Diagram (ERD)



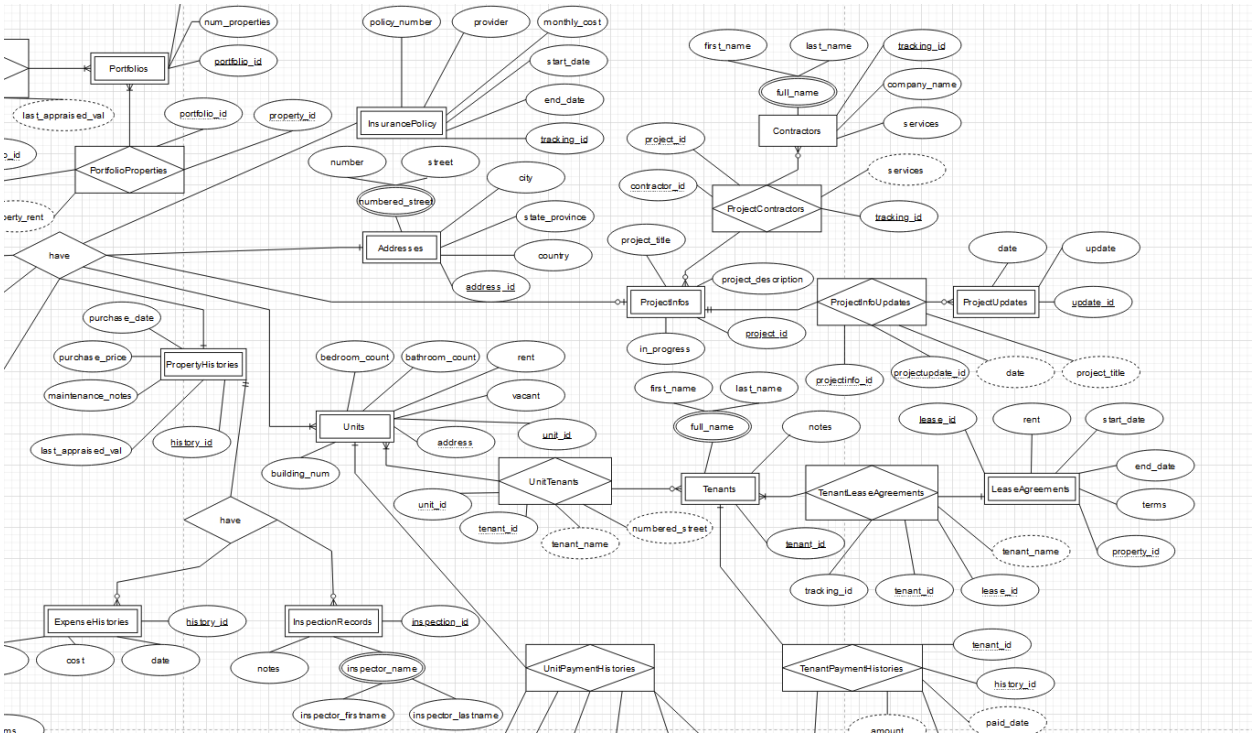
Top left:



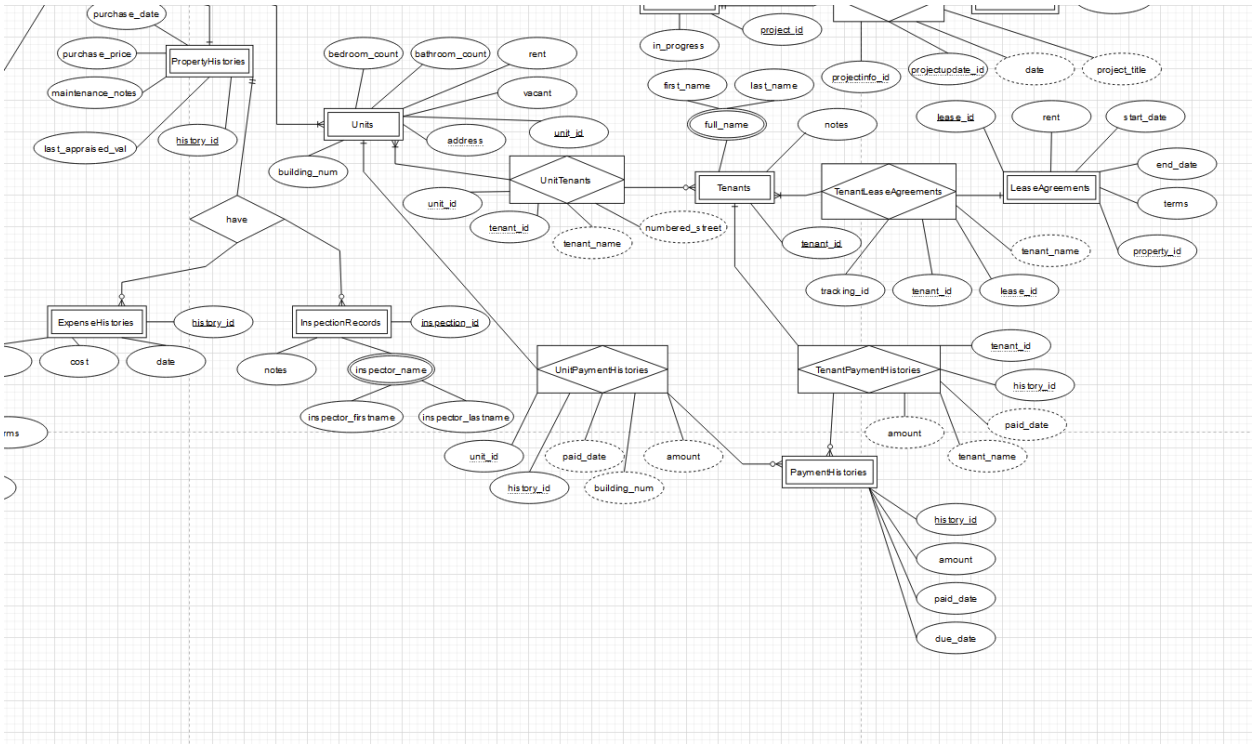
Bottom left:



Top right:



Bottom right:



Section VII: Entity Set Description

1. Users (Strong)
    - tracking_id: strong key, numeric
    - lastseen_date: date
    - firstseen_date: date

2. RegisteredUsers (Weak)
    - full_name: alphanumeric, autogenerated
    - first_name: alphanumeric
    - last_name: alphanumeric
    - password: alphanumeric
    - email: alphanumeric, composite primary key
    - tracking_id: numeric, composite primary key

3. Portfolios (Weak)
    - last_appraised_val: numeric
    - num_properties: numeric
    - portfolio_id: numeric, primary key

4. Properties (Weak)
    - target_arv: numeric
    - num_units: numeric
    - total_rent: numeric, derived
    - monthly_capex: numeric
    - bedroom_count: numeric
    - bathroom_count: numeric
    - sqft: numeric
    - lot_size: numeric
    - property_id: numeric, primary key

5. Addresses (Weak)
    - numbered_street: alphanumeric, autogenerated
    - number: numeric
    - city: alphanumeric
    - state_province: alphanumeric
    - country: alphanumeric
    - address_id: numeric, primary key

6. Tenants (Weak)
    - full_name: alphanumeric, autogenerated
    - first_name: alphanumeric
    - last_name: alphanumeric

- notes: alphanumeric
- tenant_id: numeric, primary key

7. PropertyHistories (Weak)
    - purchase_date: date, composite
    - purchase_price: numeric
    - maintenance_notes: alphanumeric
    - last_appraised_val: numeric
    - history_id: numeric, primary key
    - property_id: numeric, foreign key (references the Properties entity set)

8. Units (Weak)
    - bedroom_count: numeric
    - bathroom_count: numeric
    - rent: numeric
    - vacant: tinyint (boolean)
    - unit_id: numeric, primary key
    - address: numeric, foreign key (references the Addresses entity set)
    - building_num: numeric

9. PaymentHistories (Weak)
    - history_id: numeric, primary key
    - amount: numeric
    - paid_date: date, composite
    - due_date: date, composite

10. ExpenseHistories (Weak)
    - history_id: numeric, foreign key (references the PropertyHistories entity set)
    - cost: numeric
    - date: date, composite
    - label: alphanumeric
    - expense_id: numeric, primary key

11. ProjectInfos (Weak)
    - project_title: alphanumeric
    - project_description: alphanumeric
    - project_id: numeric, primary key
    - in_progress: tinyint (boolean)

12. InsurancePolicies (Weak)
    - policy_number: numeric
    - provider: alphanumeric
    - monthly_cost: numeric
    - start_date: date, composite

- end_date: date, composite
- tracking_id: numeric, primary key

13. Contractors (Strong)
    - full_name: alphanumeric, autogenerated
    - first_name: alphanumeric
    - last_name: alphanumeric
    - tracking_id: numeric, primary key
    - company_name: alphanumeric
    - services: alphanumeric

14. InspectionRecords (Weak)
    - inspection_id: numeric, primary key
    - inspector_name: alphanumeric, autogenerated
    - inspector_firstname: alphanumeric
    - inspector_lastname: alphanumeric
    - notes: alphanumeric

15. LeaseAgreements (Weak)
    - lease_id: numeric, primary key
    - rent: numeric
    - start_date: date, composite
    - end_date: date, composite
    - terms: alphanumeric
    - property_id: numeric, foreign key (references Properties entity set)

16. TaxRecords (Weak)
    - tracking_id: numeric, primary key
    - property_id: numeric, foreign key (references Properties entity set)
    - payment_date: date, composite
    - due_date: date, composite
    - amount_paid: numeric
    - year: numeric

17. Mortgages (Weak)
    - tracking_id: numeric, primary key
    - lender_name: alphanumeric
    - principal_balance: numeric
    - interest_rate: numeric
    - monthly_payment: numeric
    - start_date: date, composite
    - end_date: date, composite
    - property_id: numeric, foreign key (references Properties entity set)

- terms: alphanumeric

18. UserPortfolios (Associative)
    - tracking_id: numeric, primary key
    - last_appraised_val: numeric, derived
    - portfolio_id: numeric, composite primary key and foreign key (references Portfolios entity set)
    - user_id: numeric, composite primary key and foreign key (references RegisteredUsers entity set)

19. ProjectUpdates (Weak)
    - date: date, composite
    - update: alphanumeric
    - update_id: numeric, primary key

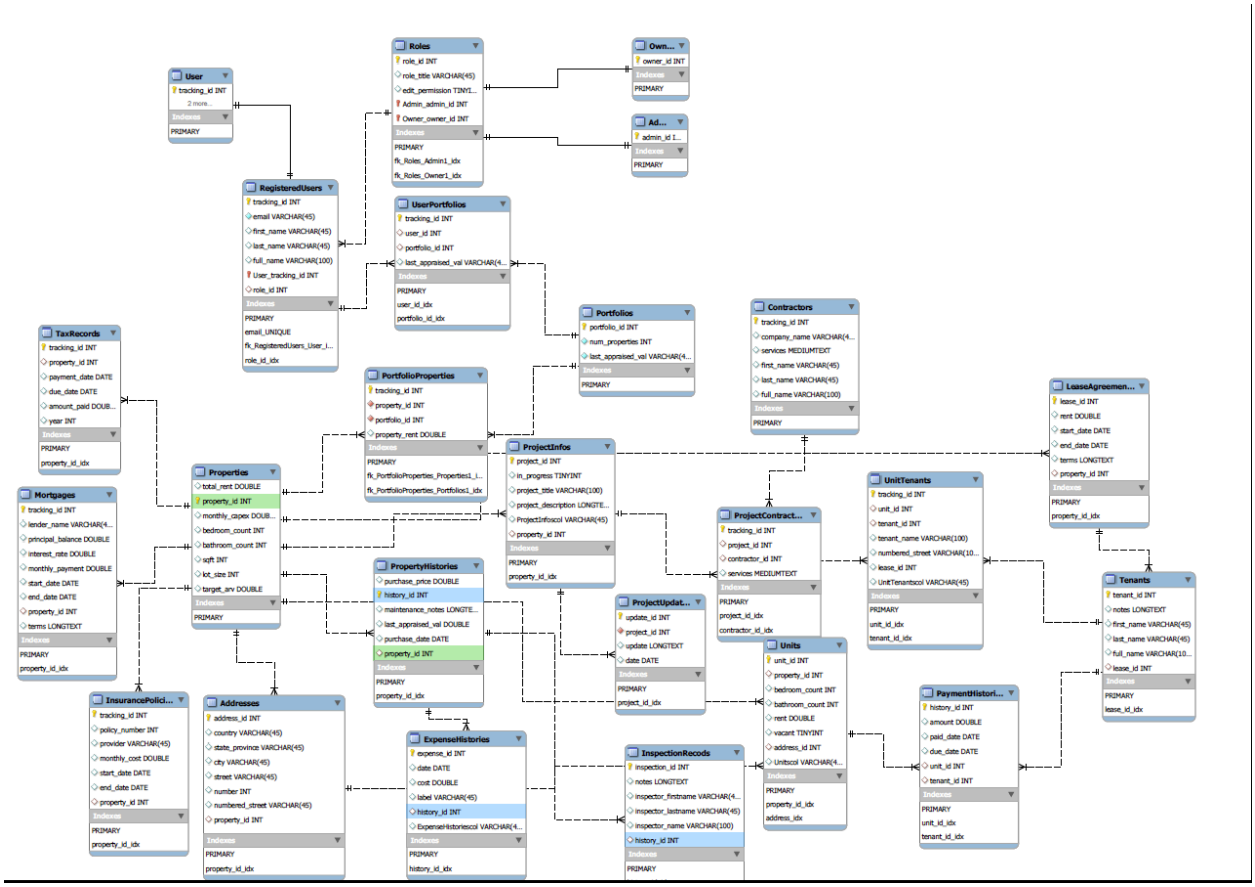
20. ProjectContractors (Associative)
    - project_id: numeric, composite primary key and foreign key (references ProjectInfos entity set)
    - contractor_id: numeric, composite primary key and foreign key (references Contractors entity set)
    - services: alphanumeric, derived
    - tracking_id: numeric, primary key

21. Roles
    - role_id: numeric, primary key
    - role_title: alphanumeric
    - edit_permission: boolean

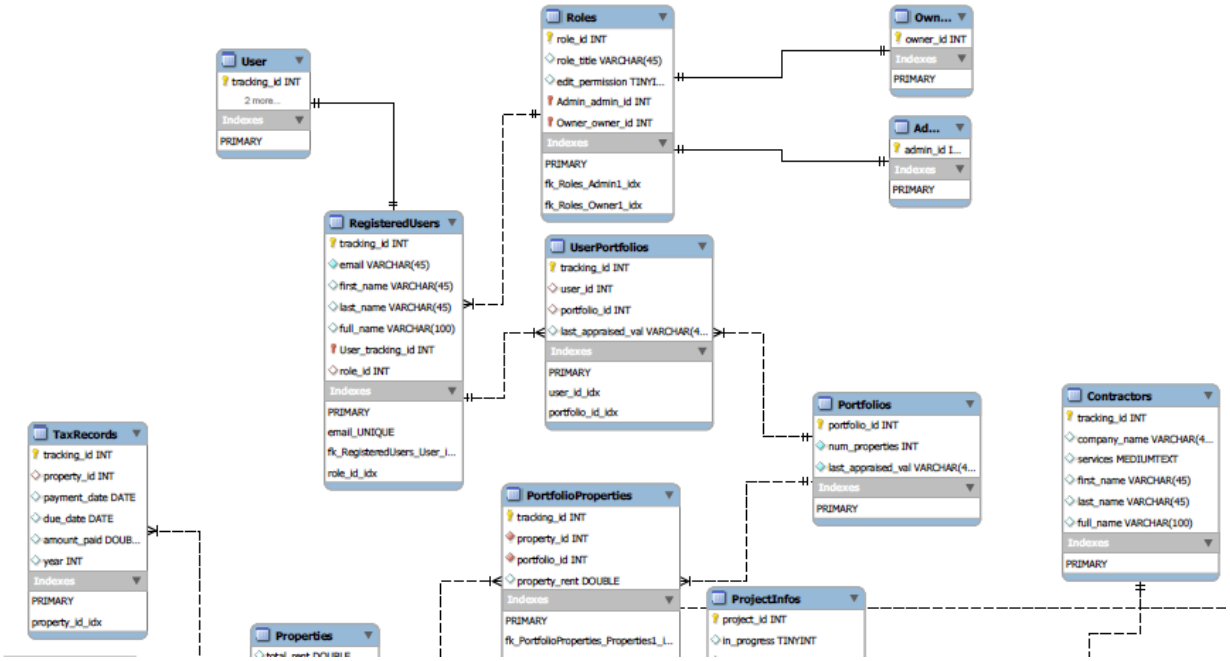
22. UnitTenants (Associative)
    - unit_id: numeric, composite primary key and foreign key (references Units entity set)
    - tenant_id: numeric, composite primary key and foreign key (References Tenants entity set)
    - tenant_name: alphanumeric, derived
    - numbered_street: alphanumeric, derived

23. PortfolioProperties (Associative)
    - portfolio_id: numeric, composite primary key and foreign key (references Portfolios entity set)
    - property_id: numeric, composite primary key and foreign key (references Properties entity set)
    - property_rent: numeric, derived

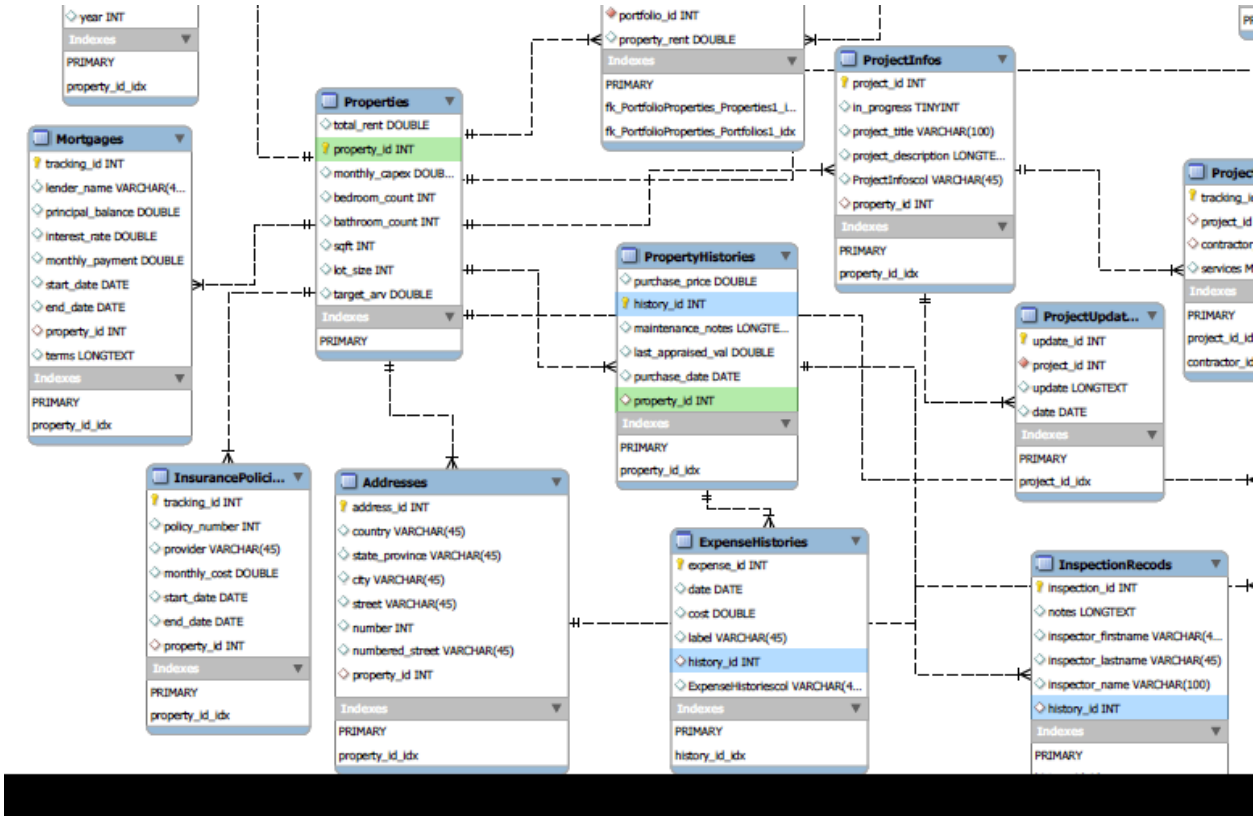
Section VIII: Enhanced Entity-Relationship (EER) Diagram (EER)



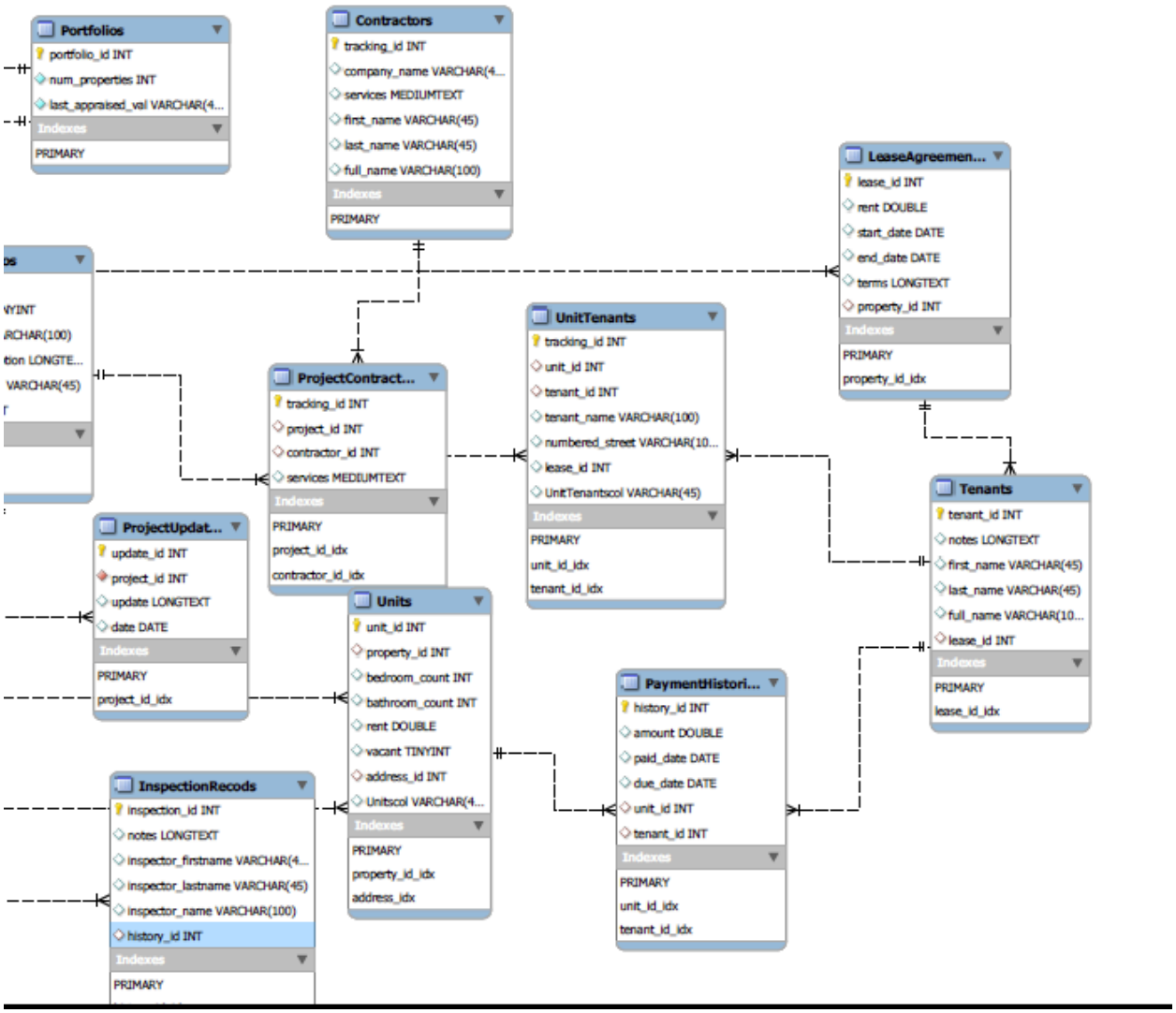
Top:



Bottom left:



Bottom right:



Section IX: Normalization Techniques Used

Each table has a tracking ID as a primary key, and when it needs to refer to another table, has the tracking ID in the referenced table as a foreign key within itself. No groups are repeated as well, so this achieves 1NF.

Most tables attempt not to have non-key attributes depend on other entities for their information in order to achieve 2NF, however, this was unavoidable with some. For example, the total_rent attribute in Properties will need to pull from LeaseAgreements, or associative tables will need to pull from the tables they are connecting. For example, UnitTenants, an associative table, contains attributes for the tenant's name and the numbered street of the property - in this way, when this table is referenced, it'll need to complete tracing the tenant's ID to find their name, and trace the unit id to the address id of the unit in order to get the address. This will be done using generation, so that the system doesn't have to trace it every time - rather, the numbered address will be stored with the persistent keyword.

Lastly, I did my best to avoid transitive dependencies with non-keys by keeping communication between foreign keys to get any other information required, and storing it as persistent in order to achieve 3NF where possible.