This document outlines the schema of the GURS database in NOSQL format, represented using JSON.

The database comprises the following collections:

- users
- properties
- portfolios
- transactions
- leases
- financial_reports

The decision to organize data into distinct collections was made to enhance document readability, ensure accuracy, and manage size effectively. By segregating the data into separate collections and linking them through relevant indexes, we minimize data redundancy and simplify user queries. Moreover, given the database's frequent updates and occasional heavy data loads, maintaining separate collections is crucial for optimal management.

Here are examples of sample documents for each collection after proper setup and data loading (look at instructions to see how to set up data if adding in bulk):

Below these are sample documents with filled fields and further explanation for effectively building/adding more documents to the database.

users:

```
{
  "_id": { "$oid": "" },
  "user_id": null,
  "first_name": "",
  "last_name": "",
  "email": "",
  "last_login": "",
  "password_hash": "",
  "contact_details": {
    "phone": "",
    "address": ""
  },
  "portfolios": []
}
```

properties:


```
{
  "_id": { "$oid": "" },
  "property_id": "",
  "portfolio_id": "",
  "value": null,
  "address": "",
  "acquisition_date": "",
  "square_footage": null,
  "property_type": {
    "_id": null,
    "name": ""
  },
  "leases": [],
  "financial_reports": []
}
```


transactions:


```
{
  "_id": { "$oid": "" },
  "transaction_id": null,
  "portfolio_id": "",
  "amount": null,
  "transaction_date": "",
  "description": "",
  "transaction_type": {
    "_id": null,
    "name": ""
  }
}
```

leases:


```
{
  "_id": { "$oid": "" },
  "lease_id": "",
  "property_id": "",
  "lease_start_date": "",
  "lease_end_date": "",
  "rent_amount": null,
  "renewal_date": "",
  "summary": "",
  "tenants": {
    "emails": []
  }
}
```


portfolios:


```
{
  "_id": { "$oid": "" },
  "portfolio_id": "",
  "user_id": null,
  "name": "",
  "total_value": null,
  "status": null,
  "properties": []
}
```

financial_reports:


```
{
  "_id": { "$oid": "" },
  "report_id": "",
  "property_id": "",
  "period_end_date": "",
  "period_start_date": "",
  "expenses": null,
  "income": null,
  "report_type": {
    "_id": null,
    "name": ""
  },
  "report_summary": ""
}
```

users:


```
{
  "_id": {
    "$oid": "66134a8be10537e70a816a48"
  },
  "user_id": 5,
  "first_name": "Audrye",
  "last_name": "Dalli",
  "email": "adalli4@etsy.com",
  "last_login": "2013-04-18",
  "password_hash":
"$2a$04$QiG8/.2CNZQGG.BVLDCaS.zy6DJsCXciwzGqBsJCmKeRwYWWZucEm",
  "contact_details": {
    "phone": "801-515-9321",
    "address": "70 Charing Cross Hill"
  },
  "portfolios": [
    "portfolio7444",
    "portfolio3706"
  ]
}
```

properties:


```
{
 "_id": {
  "$oid": "66134c30f8c5c6a3286546ce"
 },
 "property_id": "property16659243",
 "portfolio_id": "portfolio6169",
 "value": 63054545,
 "address": "1452 Carpenter Point",
 "acquisition_date": "2014-08-02",
 "square_footage": 19598,
 "property_type": {
  "_id": 3,
  "name": "vacant_land"
 },
 "leases": [
  "lease23815",
  "lease62253"
 ],
 "financial_reports": [
  "report7236541",
  "report1048124"
 ]
}
```


transactions:


```
{
 "_id": {
  "$oid": "66134c59f8c5c6a328654ea0"
 },
 "transaction_id": 6,
 "portfolio_id": "portfolio3668",
 "amount": 191452,
 "transaction_date": "2000-08-04",
 "description": "Inverse intangible system engine",
 "transaction_type": {
  "_id": 1,
  "name": "Investment Deposit"
 }
}
```

leases:


```
{
 "_id": {
  "$oid": "66134c6cf8c5c6a328655286"
 },
 "lease_id": "lease52445",
 "property_id": "property16555660",
 "lease_start_date": "2020-02-18",
 "lease_end_date": "3/21/2003",
 "rent_amount": 5095,
 "renewal_date": "1/7/2013",
 "summary": "Ameliorated local budgetary management",
 "tenants": {
  "emails": [
   "egonthard1@nytimes.com",
   "wnaile1@indiatimes.com",
   "ggarey1@ezinearticles.com",
   "tcakes1@harvard.edu"
  ]
 }
}
```


properties:


```
{
 "_id": {
  "$oid": "66134c19f8c5c6a3286542e1"
 },
 "portfolio_id": "portfolio5177",
 "user_id": 880,
 "name": "Bigtax",
 "total_value": 133877932,
 "status": false,
 "properties": [
  "property35145109",
  "property11898153"
 ]
}
```

financial_reports:


```
{
 "_id": {
  "$oid": "66134c3ef8c5c6a328654ab1"
 },
 "report_id": "report7478318",
 "property_id": "property6976706",
 "period_end_date": "2021-09-28",
 "period_start_date": "7/9/2017",
 "expenses": 666487,
 "income": 977501,
 "report_type": {
  "_id": 4,
  "name": "Hyatt Hotels Corporation"
 },
 "report_summary": "Future-proofed 5th generation access"
}
```

Instructions for potential engineers:

To ensure that our MongoDB schema is well-designed and conducive to efficient queries, I'll explain the relationships and references among the various collections we plan to use. Here's how the collections interrelate:

1. Users Collection
- Description: This collection stores data related to users, including personal information and authentication data.
- References:
  - Portfolios: A user can own multiple portfolios. This relationship is represented by an array of `portfolio_id` values stored in the `portfolios` field of the `Users` document.

2. Portfolios Collection
- Description: Contains information about investment portfolios, which are collections of properties.
- References:
  - Users: Each portfolio is linked to a single user via the `user_id` field. This establishes ownership.
  - Properties: Portfolios reference multiple properties through an array of `property_id` values in the `properties` field. This linkage allows us to quickly find all properties belonging to a particular portfolio.
  - Transactions: Although not directly stored within the portfolio documents, transactions related to financial activities within the portfolios can be linked through the `portfolio_id` present in each transaction.

3. Properties Collection
- Description: Holds data about physical properties, such as location, value, and type.
- References:
  - Portfolios: Each property is associated with a portfolio through the `portfolio_id` field.
  - Leases: Properties can have multiple leases associated with them, stored as an array of `lease_id` values.
  - Financial Reports: Similarly, financial activities related to each property are documented and linked via an array of `report_id` values.

4. Transactions Collection
- Description: Records financial transactions, each linked to a specific portfolio.
- References:
  - Portfolios: Each transaction references a portfolio, indicated by the `portfolio_id`. This helps in aggregating all transactions for a given portfolio.

5. Leases Collection
- Description: Manages lease agreements for properties, including terms and tenant information.
- References:
  - Properties: Each lease is associated with a specific property, referenced by the `property_id`. This connection is crucial for managing and querying lease terms and tenant details specific to a property.

6. Financial Reports Collection
- Description: Provides a detailed account of the financial performance of each property, including income and expenses.
- References:
  - Properties: Similar to leases, each financial report is tied to a property via the `property_id`. This association allows for easy access to financial data relevant to particular properties.

Design Considerations

Normalization vs. Denormalization: Our design moderately normalizes relationships (using references) rather than embedding documents. This approach balances query flexibility with data integrity and avoids excessive duplication.

Referential Integrity: We rely on application logic to maintain referential integrity, as MongoDB does not enforce foreign key constraints.

Indexing: Important to index reference fields (user_id, portfolio_id, property_id, etc.) to optimize query performance, especially for lookups and aggregations that span multiple collections.

Aggregation Framework: Extensively used for joining data across collections, especially for compiling comprehensive reports or snapshots of portfolios, properties, and their financial standings.

This schema design allows for scalable data management practices, supporting complex queries and operations vital for real estate and investment platforms. It provides a clear roadmap for implementing the database structure, ensuring each collection serves a distinct purpose while maintaining a coherent relationship model.