Global Startup Ecosystem:

Classes:

- **Startup**: Attributes: **startupID**, **name**, **category**, **fundingTotal**, **status**.
- **Investor**: Attributes: **investorID**, **name**, **investorType**.
- **FundingRound**: Attributes: **roundID**, **fundingRoundType**, **fundingDate**.
- **Venue**: Attributes:**venueID**, **venueName**, location.
- **Founder**: Attributes: **founderID**, **founderName, birthDate, nationality**.
- **Relationships**:
- Many-to-many between **Startup** and **Investor** through **FundingRound**.
- One-to-many between **Venue** and **Startup**

Nouns

Verbs

Rules

- A Venue can host multiple funding rounds but a funding round can be hosted at a single Venue.
- Each Startup may receive investments from multiple **Investors** through different Funding Rounds.
- An **Investor** can invest in multiple **Startups** and participate in various **Funding Rounds**.
- **Investors** can be categorized into **Individual Investors** and **Institutional Investors**.
- Each **Funding Round** is associated with one **Startup** and one **Investor**.
- A Founder can be associated with multiple Startups.
- A Startup can have multiple Founders.

Nouns

- Venue
- Startup
- Funding Rounds
- Investor
- Individual Investor
- Institutional Investor
- Founder

Verbs

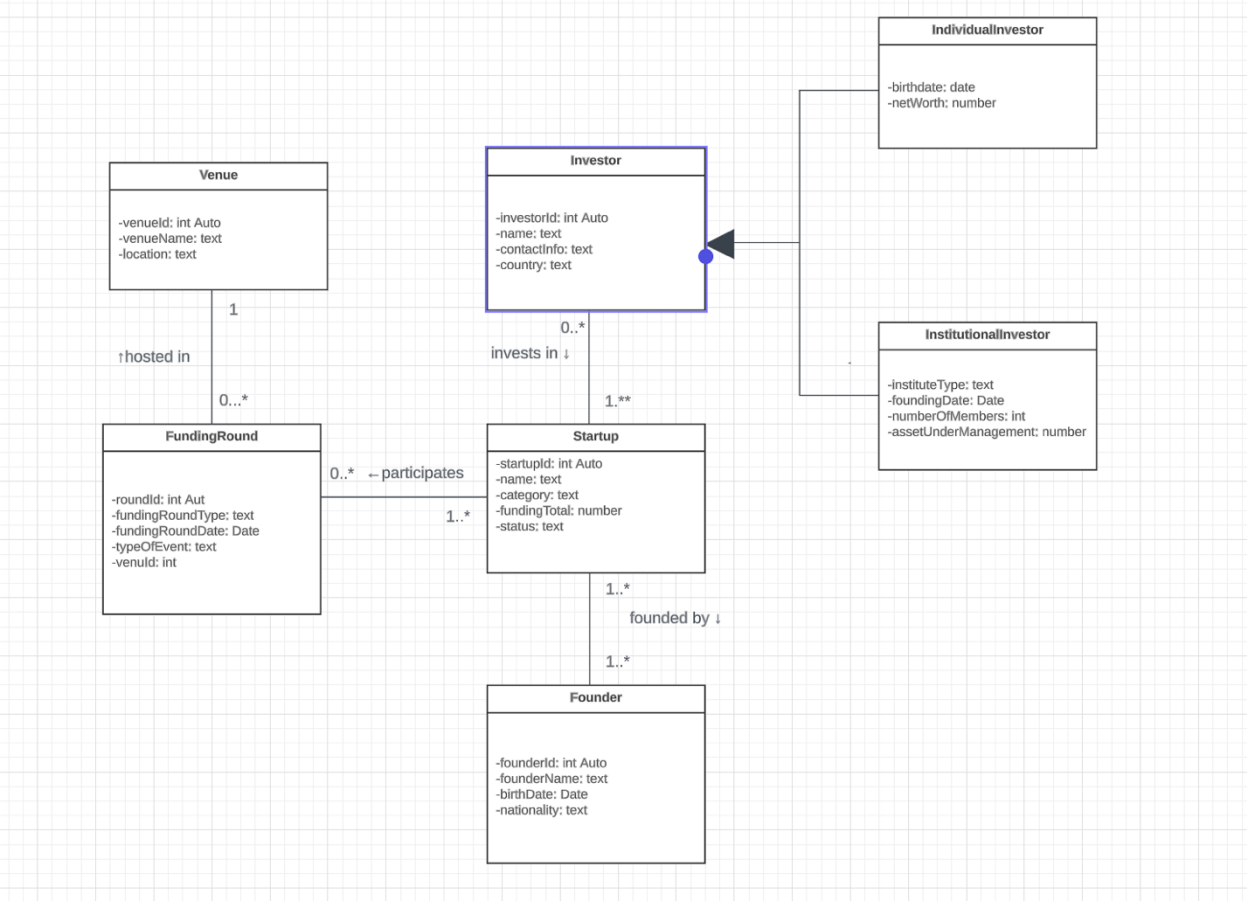
- Hosts
- Invests
- Participates
- Associated
- Receive
- Categorized

List of Possible Actions

- Register a new Startup
- Update Startup details
- Delete a Startup
- Add a new Investor
- Update Investor details
- Delete an Investor
- Record a new Funding Round
- Update Funding Round details
- Delete a Funding Round
- View all Startups from a specific Country
- View all Funding Rounds for a particular Startup
- View all Investments made by an Investor
- Analyze Investor preferences by Country
- Determine Startup categories preferred by Investors from a specific Country
- Analyze the funding received by a certain type of startups
- Determine the type of startups receiving the most number amount of funding.

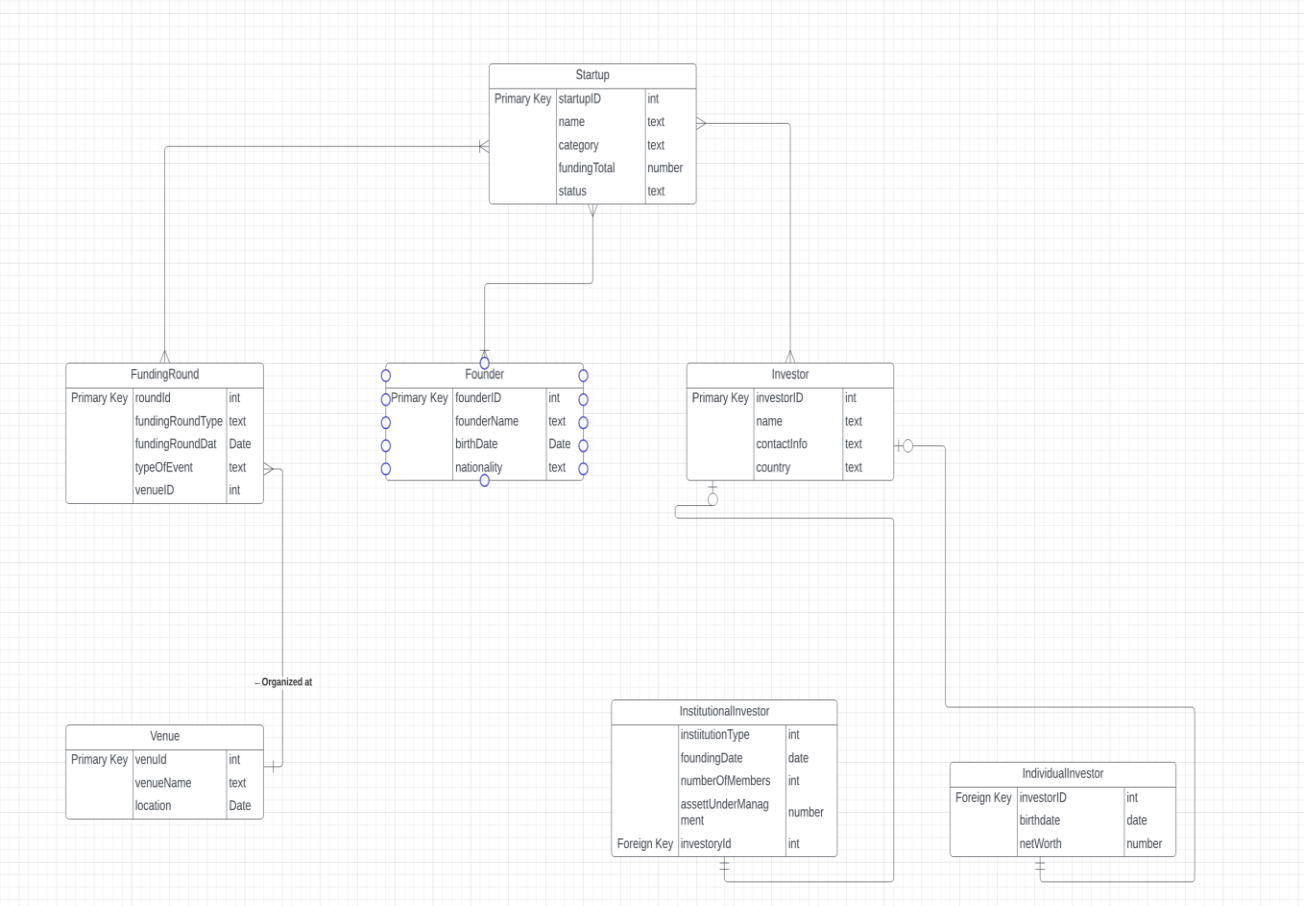
UML Diagram

[Link to UML diagram]



ERD Diagram

[Link to ERD Diagram]



Mongo Documents

1. Startup Collection

Contains startup details.

An array of founderIDs references the Founder collection.

An array of investorIDs references the Investor collection.

An array of fundingRoundIDs references the FundingRound collection.

Example JSON

```
{
  "startupDetails": { /* ... */ },
  "founders": [/* array of ObjectIds referencing Founder documents */],
  "investors": [/* array of ObjectIds referencing Investor documents */],
  "fundingRounds": [/* array of ObjectIds referencing FundingRound documents */]
}
```

2.Investor Collection

Stores common details for all investors, using a discriminator field to distinguish between individual and institutional investors.

An array of startupIDs to represent the many-to-many relationship with startups.

Example JSON

```
{
  "commonDetails": { /* ... */ },
  "investorType": "Individual" | "Institutional",
  "individualDetails": { /* ... */ }, // Exists only if investorType is "Individual"
  "institutionalDetails": { /* ... */ }, // Exists only if investorType is "Institutional"
  "startups": [/* array of ObjectIds referencing Startup documents */]
```

}

3.FundingRound Collection

Contains details specific to each funding round.

An object for venue details, potentially a subdocument if the venue data is small and not frequently updated on its own.

An array of participatingStartupIDs references the Startup collection.

Example JSON

{

"fundingRoundDetails": { /* ... */ },

"venue": { /* subdocument with venue details */ },

"participatingStartups": [/* array of ObjectIds referencing Startup documents */]

}

4.Founder Collection

Founder details are stored in this collection.

Could be referenced by founderIDs within the Startup collection.

json

Copy code

{

"founderDetails": { /* ... */ }

}

Note: For the Investor collection, which is a generalization of individual and institutional investors, the discriminator field (investorType) helps to manage this generalization by indicating the type of investor each document represents.

Redis Database

For this project I will be using redis for:

1. Investments of every investor

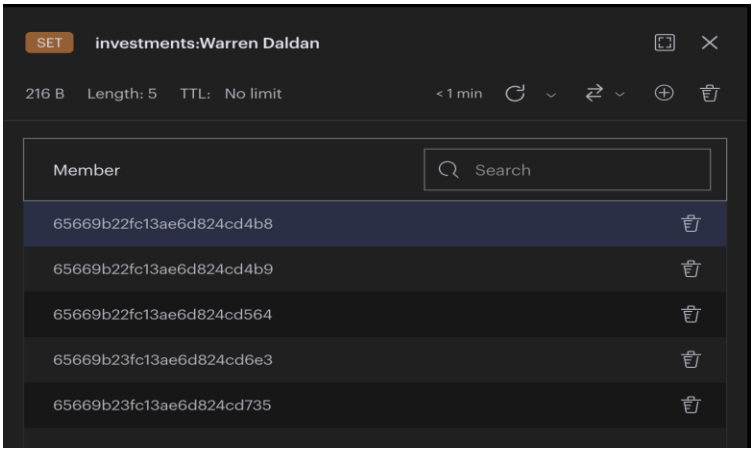
The above image shows my “Investor” collection in mongoDB.

Here object ids of each startup is stored in array called “startup”. The object ids are the ids of the data in “Startup” collections which consists of details for each startup.

In my redis database I will be using **“SETS”** datatype to store.

Key design: investments:Investor_name

Example:



CRUD operations

Create: node script in db/redis/investsHashMap.js creates this datastructure, where it create keys and add values for each investor

Update: the investor can select startup to invests and its object id will be added to the set in the redis databse

Delete: The investor can delete any startup from his investments

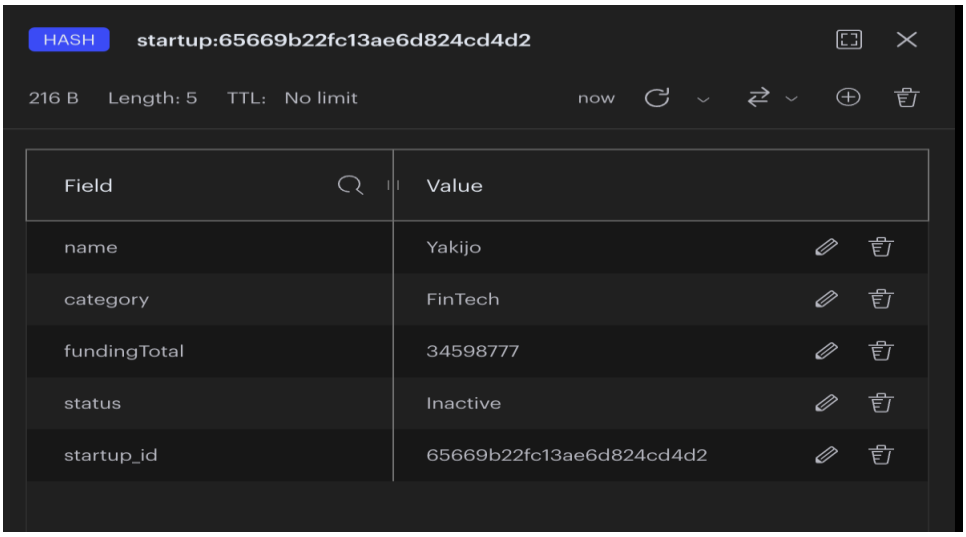
Read: The investment page that displays the list of invested startups for each investor.

2. Startup Details

```
_id: ObjectId('000000000000000000000001')
startupDetails: Object
    name: "TechXYZ"
    category: "Software"
    fundingTotal: "1000000"
    status: "active"
founders: Array (4)
    0: ObjectId('000000000000000000000001')
    1: ObjectId('6566a1aabc09b3e8747a5246')
    2: ObjectId('6566a1aabc09b3e8747a4f38')
    3: ObjectId('6566a1aabc09b3e8747a507b')
investors: Array (6)
    0: ObjectId('000000000000000000000001')
    1: ObjectId('6566a135bc09b3e8747a4c57')
    2: ObjectId('6566a0aabc09b3e8747a4947')
    3: ObjectId('6566a0aabc09b3e8747a4918')
    4: ObjectId('6566a0aabc09b3e8747a49c4')
    5: ObjectId('6566a135bc09b3e8747a4e0d')
fundingRound: Array (6)
    0: ObjectId('000000000000000000000001')
    1: ObjectId('6566a585bc09b3e8747a5605')
    2: ObjectId('6566a585bc09b3e8747a53ed')
    3: ObjectId('6566a585bc09b3e8747a53f5')
```


The above image shows a data in the “Startup” collection. I have used “hash” data type of redis to map the objects ids of startups to the startup data.

Key design: startup:Object_id



CRUD Operations:

Create: in db/redis/startupHash.js creates hashes for each startup from the mongodb database

Read: In the investments page the startup is displayed

Overview of the redis database: