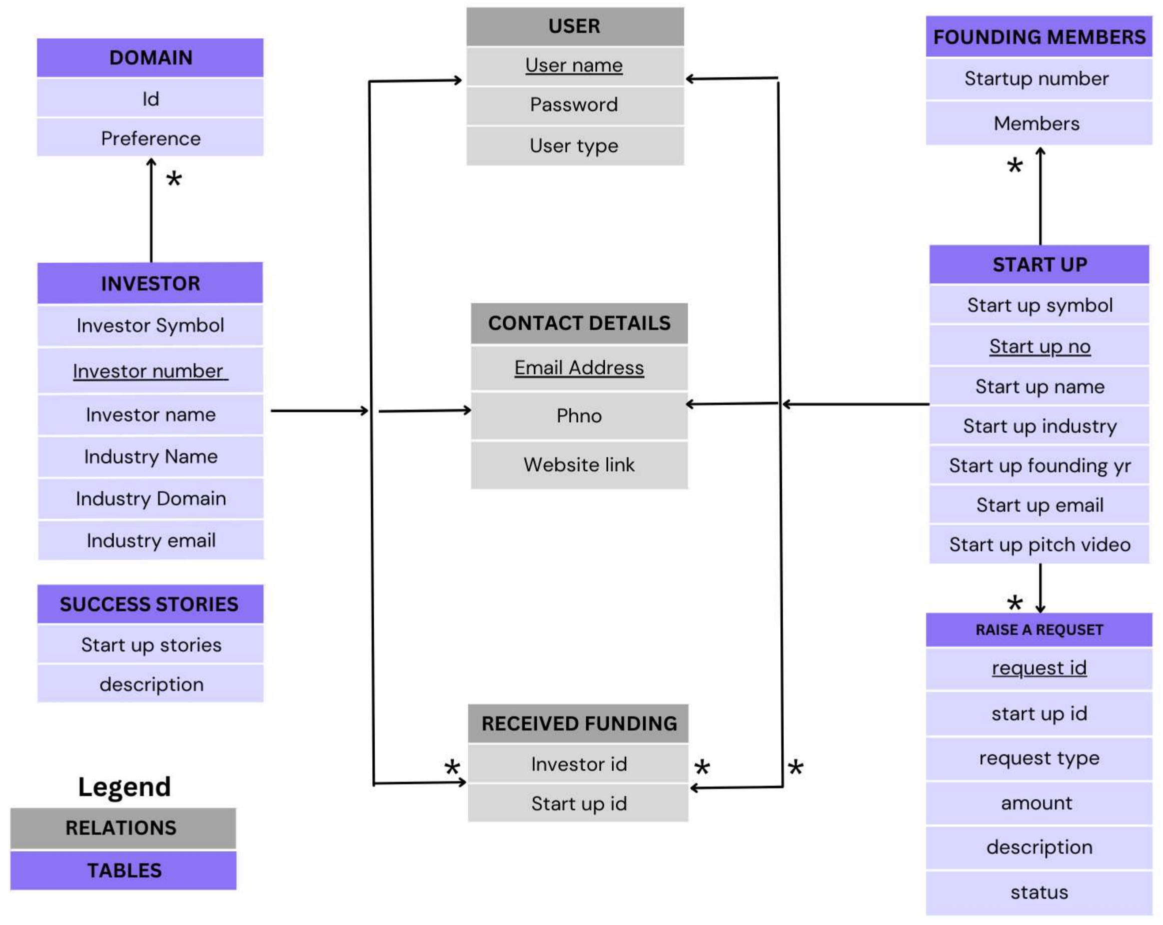
ERD-



Normalization of the database-

Tables

Start up table-

Before normalization-

Attributes -
1)startup symbol
2) startup number

3) name (varchar)
4)description (text)
5)industry (varchar (20))
8) contact info (bigInt)
9) email id
10) website link
11) funding requirements
12) pitch video (blob)
13)founding members
14)Password (Varchar)

Functional Dependencies:

- Investor ID → Name, Investment Preferences, Phone Number, Email, Website, Password
- Email → Phone Number, Website

Startup Table:

- Startup symbol
- Startup number (primary key)
- Startup description
- Start up Industry
- Start up founding year
- Pitch Video
- User id

Founding members: (composite primary key)

- Startup ID
- Founding member

Contact Table:

- Startup ID (Foreign Key referencing Startup Table)
- Contact Info
- Email ID
- Website Link

Investor table-

1) Investor id (primary key , varchar (10))
2) Name (varchar (100))
3) Investment preferences (text)
4) PhoneNo
5) Email
6) Website
7) Password (Varchar)

Functional Dependencies:

- Investor ID → Name, Investment Preferences, Phone Number, Email, Website, Password
- Email → Name, Investment Preferences, Phone Number, Website, Password

Investor Table: (composite primary key of Num and symbol)

- InvestorNumber
- InvestorSymbol
- Investor name
- Industry name
- Industry Domain
- Investor Email (foreign key)
- User id

Contact Table:

- Email ID (foreign key investor table)
- Contact Info (BIGINT)
- Website Link

Relations-

User-

- User name
- Password
- User type

Received funding-

- Investor id
- Start up id

Sql queries-

Create table commands -

User table

```
CREATE TABLE User (
   UserName VARCHAR(64) PRIMARY KEY,
   Password VARCHAR(100) NOT NULL,
   UserType ENUM('startup', 'investor') NOT NULL
);
```

Contact Details table

```
CREATE TABLE contactDetails (
     EmailAddress VARCHAR(100) PRIMARY KEY,
     Phno BIGINT,
     WebsiteLink VARCHAR(100)
  );
```

Startup Table

```
CREATE TABLE StartUp (
   Startup_symbol CHAR(1) DEFAULT 'S',
  Startup_no INT NOT NULL AUTO_INCREMENT,
  Startup_name VARCHAR(70),
  Startup_description TEXT,
   Startup_industry VARCHAR(100),
   Startup_founding_year DATE,
    Startup_email VARCHAR(100),
    PitchVideo LONGBLOB,
    PRIMARY KEY (Startup_no),
    UNIQUE KEY (Startup_symbol, Startup_no),
    FOREIGN KEY (Startup_email) REFERENCES contactDetails(EmailAddress)
   );
ALTER TABLE StartUp
ADD COLUMN User_ID VARCHAR(64),
ADD FOREIGN KEY (User_ID) REFERENCES User(UserName);
```

Founding members table

```
CREATE TABLE FoundingMembers(
    s_symbol VARCHAR(1),
    s_no INT ,
    Members VARCHAR(100) ,
    FOREIGN KEY(s_symbol,s_no) REFERENCES StartUp(Startup_symbol,Startup_no)
);
```

Investor table

```
CREATE TABLE Investor (
     InvestorSymbol VARCHAR(1) DEFAULT 'I', -- Corrected the quotation marks
     InvestorNumber INT NOT NULL AUTO_INCREMENT,
     InvestorName VARCHAR(100),
     IndustryName VARCHAR(100),
     InvestorEmail VARCHAR(255),
     IndustryDomain VARCHAR(100),
     PRIMARY KEY (InvestorNumber),
     UNIQUE KEY (InvestorSymbol, InvestorNumber),
     FOREIGN KEY (InvestorEmail) REFERENCES contactDetails(EmailAddress)
    );
ALTER TABLE Investor
ADD COLUMN User_ID VARCHAR(64),
ADD FOREIGN KEY (User_ID) REFERENCES User(UserName);
```

Domain table

```
CREATE TABLE Domain (
Investor_ID int ,
Preference VARCHAR(50),
PRIMARY KEY(Investor_ID,Preference) );
```

Request table

```
CREATE TABLE Request (
   RequestID INT AUTO_INCREMENT PRIMARY KEY,
   StartupID INT NOT NULL,
   RequestType VARCHAR(255),
   Amount DECIMAL(10, 2),
   Description TEXT,
   Status VARCHAR(50),
   FOREIGN KEY (StartupID) REFERENCES Startup(Startup_no)
);
```

Received Funding table

```
CREATE TABLE ReceivedFunding(

Startup_no  int,

InvestorNumber int ,

FOREIGN KEY (Startup_no ) REFERENCES Startup(Startup_no),
FOREIGN KEY (InvestorNumber) REFERENCES Investor(InvestorNumber)

);
```

Log table

```
CREATE TABLE log (

   Action VARCHAR(200),

  PerformedAt DATETIME DEFAULT Current_TimeStamp

  );
```

Success Stories table -

```
CREATE TABLE SuccessStories (
   startup_name VARCHAR(255),
   description TEXT
    );
```

Create view commands-

Startup view

```
CREATE VIEW startupView AS
   SELECT Startup_name,Startup_industry,Startup_email
    FROM startup;
```

Investor view

```
CREATE VIEW InvestorView as
    SELECT InvestorName,InvestorEmail
    from Investor;
```

Create trigger commands

```
DELIMITER //
CREATE TRIGGER withdraw_trigger
   BEFORE UPDATE ON request
   FOR EACH ROW
   BEGIN
     IF OLD.Status = 'R' AND NEW.Status = 'W' THEN
```

```
  INSERT INTO oldrequest (RequestID, StartupID, RequestType, Amount, Description, Status)

    VALUES (OLD.RequestID, OLD.StartupID, OLD.RequestType, OLD.Amount,
OLD.Description, OLD.Status);

   END IF;

   END;

    //
```

Oldrequest table

```
Create table OldRequest(

  RequestID int ,

  StartupID int ,

  RequestType varchar(225),

  Amount decimal(10,2),

  Description text,

  Status varchar(50),

   Primary Key(RequestID),

  Foreign key(StartupID) references Startup(Startup_no)

   );
```

Future Enhancements-

- Integration with external APIs for real-time market analysis and investor profiling.
- Implementation of advanced matching algorithms for improved precision in connecting startups with investors.
- Enhanced user interface design for a more intuitive user experience.
- Integration with payment gateways to facilitate seamless fund transactions between startups and investors.

Conclusion-
StartUpSupport provides a robust platform for startups to raise funds and investors to discover promising investment opportunities. By leveraging technology and data-driven matching algorithms, the application aims to foster meaningful collaborations and drive innovation in the startup ecosystem.