model User {

id Int @id @default(autoincrement())

firstName String

lastName String

email String @unique

username String @unique

password String

role Role

institution Institution? @relation(fields: [institutionId], references: [id])

institutionId Int?

isActive Boolean @default(true) // Field to track if the user is active or deactivated

subscription UserSubscription? @relation(fields: [subscriptionId], references: [id])

subscriptionId Int?

@@index([email])

@@index([username])

// Relationships

uploadedContent Content[] @relation("Uploader")

reviewedContent Content[] @relation("Reviewer")

notifications Notification[]

comments Comment[]

accessRequests AccessRequest[]

commentReports CommentReport[]

feedbacks Feedback[]

}

model Feedback {

id Int @id @default(autoincrement())

content String

createdAt DateTime @default(now())

userId Int

resolved Boolean @default(false)

// Relationships

user User @relation(fields: [userId], references: [id])

}

model SubscriptionPlan {

id Int @id @default(autoincrement())

name String

description String

price Float

duration DurationType

createdAt DateTime @default(now())

// Relationships

subscriptions UserSubscription[]

}

model UserSubscription {

id Int @id @default(autoincrement())

startDate DateTime @default(now())

endDate DateTime

isActive Boolean @default(true)

userId Int

planId Int

// Relationships

user User @relation(fields: [userId], references: [id])

plan SubscriptionPlan @relation(fields: [planId], references: [id])

}

model Institution {

id Int @id @default(autoincrement())

name String

address String

contactEmail String

contactPhone String

registrationStatus RegistrationStatus @default(PENDING)

admin User? @relation(fields: [adminId], references: [id])

adminId Int?

// Relationships

users User[] @relation("Institution")

content Content[]

}

model Content {

id Int @id @default(autoincrement())

title String

description String

contentType ContentType

fileUrl String

metaData Json

accessLevel AccessLevel

uploaderId Int

reviewerId Int?

institution Institution @relation(fields: [institutionId], references: [id])

institutionId Int

viewOnly Boolean @default(true) // Indicates that content is view-only

isFree Boolean @default(false) // Indicates if content can be downloaded

// Relationships

uploader User @relation("Uploader", fields: [uploaderId], references: [id])

reviewer User @relation("Reviewer", fields: [reviewerId], references: [id])

accessRequests AccessRequest[]

comments Comment[]

}

model Comment {

id Int @id @default(autoincrement())

text String

createdAt DateTime @default(now())

userId Int

contentId Int

// Relationships

user User @relation(fields: [userId], references: [id])

content Content @relation(fields: [contentId], references: [id])

reports CommentReport[]

}

model CommentReport {

id Int @id @default(autoincrement())

createdAt DateTime @default(now())

reason String

resolved Boolean @default(false)

userNotified Boolean @default(false) // Indicates if the user has been notified of the report

commentId Int

userId Int

// Relationships

comment Comment @relation(fields: [commentId], references: [id])

user User @relation(fields: [userId], references: [id])

}

model Notification {

id Int @id @default(autoincrement())

message String

isRead Boolean @default(false)

createdAt DateTime @default(now())

user User @relation(fields: [userId], references: [id])

userId Int

}

model AccessRequest {

id Int @id @default(autoincrement())

createdAt DateTime @default(now())

status RequestStatus

researcherId Int

contentId Int

researcherFile String?

// Relationships

researcher User @relation(fields: [researcherId], references: [id])

content Content @relation(fields: [contentId], references: [id])

reviewer User @relation("ReviewerAccessRequest", fields: [reviewerId], references: [id])

reviewerId Int?

}

enum DurationType {

DAILY

MONTHLY

YEARLY

}

enum Role {

PLATFORM\_ADMIN

PREMIUM\_USER

PUBLIC\_USER

RESEARCHER\_USER

INSTITUTION\_ADMIN

UPLOADER

REVIEWER

}

enum RegistrationStatus {

PENDING

APPROVED

REJECTED

}

enum ContentType {

VIDEO

BOOK

TEXT

MUSIC

PHOTO

}

enum AccessLevel {

RESEARCHER

PREMIUM\_USER

FREE

PUBLIC

}

enum RequestStatus {

PENDING

APPROVED

REJECTED

}