Database Schema Documentation

Complete database schema for the SupportDesk application using Prisma ORM with PostgreSQL.

Entity Relationship Diagram

```
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│     Account     │     │      User       │     │     Session     │
├─────────────────┤     ├─────────────────┤     ├─────────────────┤
│ id              │────►│ id (PK)         │◄────│ id              │
│ userId (FK)     │     │ name            │     │ userId (FK)     │
│ provider        │     │ email (unique)  │     │ sessionToken    │
│ providerAcctId  │     │ password        │     │ expires         │
│ access_token    │     │ role (enum)     │     └─────────────────┘
│ refresh_token   │     │ image           │
└─────────────────┘     │ createdAt       │
                        │ updatedAt       │
                        └────────┬────────┘
                                 │
                  ┌──────────────┼──────────────┐
                  │              │              │
         ┌────────▼────────┐ ┌───▼─────────┐ ┌──▼──────────────┐
         │     Ticket      │ │   Comment   │ │ TicketActivity  │
         ├─────────────────┤ ├─────────────┤ ├─────────────────┤
         │ id (PK)         │ │ id (PK)     │ │ id (PK)         │
         │ title           │ │ content     │ │ action          │
         │ description     │ │ ticketId(FK)│ │ description     │
         │ status (enum)   │◄│ userId (FK) │ │ ticketId (FK)   │
         │ priority(enum)  │ │ createdAt   │ │ userId (FK)     │
         │ category(enum)  │ │ updatedAt   │ │ createdAt       │
         │ createdById(FK) │ └─────────────┘ └─────────────────┘
         │ assignedTo(FK)  │
         │ createdAt       │
         │ updatedAt       │
         │ resolvedAt      │
         │ closedAt        │
         └─────────────────┘
```

Models

User

Core user model with role-based access control.

Field	Type	Constraints	Description
id	String	Primary Key, CUID	Unique user identifier
name	String?	Optional	User's full name
email	String	Unique, Indexed	User's email address
emailVerified	DateTime?	Optional	Email verification timestamp
image	String?	Optional	Profile image URL
password	String?	Optional	Hashed password (for credentials)
role	UserRole	Enum, Default: customer	User's role in the system
createdAt	DateTime	Auto-generated	Account creation timestamp
updatedAt	DateTime	Auto-updated	Last update timestamp

Relations:
- `accounts` → One-to-many with Account
- `sessions` → One-to-many with Session
- `ticketsCreated` → One-to-many with Ticket (as creator)
- `ticketsAssigned` → One-to-many with Ticket (as assignee)
- `comments` → One-to-many with Comment
- `activities` → One-to-many with TicketActivity

Indexes:
- `email` (unique)
- `role`

UserRole (Enum)

Defines the three access levels in the system.

Value	Description
admin	Full system access, analytics, user management
agent	Ticket management, assigned tickets
customer	Create tickets, view own tickets

Account

Stores OAuth provider account information (NextAuth).

Field	Type	Constraints	Description
id	String	Primary Key, CUID	Account identifier
userId	String	Foreign Key → User	Associated user
type	String	Required	OAuth provider type
provider	String	Required	Provider name (google, etc)
providerAccountId	String	Required	Provider's account ID
refresh_token	String?	Optional, Text	OAuth refresh token
access_token	String?	Optional, Text	OAuth access token
expires_at	Int?	Optional	Token expiration timestamp
token_type	String?	Optional	Token type
scope	String?	Optional	OAuth scopes
id_token	String?	Optional, Text	OpenID Connect ID token
session_state	String?	Optional	OAuth session state

Constraints:
- Unique on `(provider, providerAccountId)`

Indexes:
- `userId`

Session

Stores user session information (NextAuth).

Field	Type	Constraints	Description
id	String	Primary Key, CUID	Session identifier
sessionToken	String	Unique	Session token
userId	String	Foreign Key → User	Associated user
expires	DateTime	Required	Session expiration

Indexes:
- `sessionToken` (unique)
- `userId`

VerificationToken

Stores email verification tokens (NextAuth).

Field	Type	Constraints	Description
identifier	String	Part of composite key	Email or identifier
token	String	Unique, part of composite	Verification token
expires	DateTime	Required	Token expiration

Constraints:
- Unique on `(identifier, token)`

Ticket

Core support ticket model.

Field	Type	Constraints	Description
id	String	Primary Key, CUID	Ticket identifier
title	String	Required	Ticket title/subject
description	String	Required, Text	Detailed description
status	TicketStatus	Enum, Default: open	Current ticket status
priority	TicketPriority	Enum, Default: medium	Priority level
category	TicketCategory	Enum, Required	Ticket category
createdById	String	Foreign Key → User	Ticket creator
assignedToId	String?	Foreign Key → User	Assigned agent (optional)
createdAt	DateTime	Auto-generated	Creation timestamp
updatedAt	DateTime	Auto-updated	Last update timestamp
resolvedAt	DateTime?	Optional	Resolution timestamp
closedAt	DateTime?	Optional	Closure timestamp

Relations:
- `createdBy` → Many-to-one with User
- `assignedTo` → Many-to-one with User (nullable)
- `comments` → One-to-many with Comment
- `activities` → One-to-many with TicketActivity

Indexes:
- `createdById`
- `assignedToId`
- `status`
- `priority`
- `category`
- `createdAt`

TicketStatus (Enum)

Defines ticket lifecycle states.

Value	Description
open	Newly created, awaiting attention
in_progress	Being actively worked on
resolved	Solution provided, awaiting closure
closed	Completed and closed

TicketPriority (Enum)

Defines urgency levels.

Value	Description
low	Low priority, non-urgent
medium	Normal priority
high	High priority, urgent

TicketCategory (Enum)

Defines ticket categories for organization.

Value	Description
technical_issue	Technical problems or bugs
billing	Payment and billing questions
feature_request	Suggestions for new features
general_inquiry	General questions or information

Comment

Stores conversations on tickets.

Field	Type	Constraints	Description
id	String	Primary Key, CUID	Comment identifier
content	String	Required, Text	Comment text
ticketId	String	Foreign Key → Ticket	Associated ticket
userId	String	Foreign Key → User	Comment author
createdAt	DateTime	Auto-generated	Creation timestamp
updatedAt	DateTime	Auto-updated	Last update timestamp

Relations:
- `ticket` → Many-to-one with Ticket
- `user` → Many-to-one with User

Indexes:
- `ticketId`
- `userId`
- `createdAt`

TicketActivity

Audit log for ticket changes.

Field	Type	Constraints	Description
id	String	Primary Key, CUID	Activity identifier
ticketId	String	Foreign Key → Ticket	Associated ticket
userId	String	Foreign Key → User	User who made the change
action	String	Required	Action type (created, etc)
description	String	Required, Text	Description of change
createdAt	DateTime	Auto-generated	When change occurred

Relations:
- `ticket` → Many-to-one with Ticket
- `user` → Many-to-one with User

Indexes:
- `ticketId`
- `userId`
- `createdAt`

Database Migrations

This project uses Prisma's `db push` workflow for development. For production, use migrations:

```
# Create a new migration
yarn prisma migrate dev --name description_of_changes

# Apply migrations in production
yarn prisma migrate deploy
```

Seeding

The seed script (`scripts/seed.ts`) populates the database with:
- 1 admin user
- 2 agent users
- 3 customer users
- 6 sample tickets with various statuses
- Multiple comments
- Activity logs

Run seeding:

```
yarn prisma db seed
```

Performance Considerations

Indexes

The schema includes strategic indexes on:
- Foreign keys for relationship queries
- Filter fields (status, priority, category, role)
- Timestamp fields for sorting

Cascade Deletions

- Deleting a **User** cascades to their tickets, comments, and activities
- Deleting a **Ticket** cascades to its comments and activities
- Setting assigned agent to NULL when agent is deleted

Future Optimizations

- Add pagination to ticket queries
- Implement database-level full-text search
- Add composite indexes for common query patterns
- Consider archiving old closed tickets

Prisma Commands

```
# Generate Prisma Client
yarn prisma generate

# Push schema changes to database (development)
yarn prisma db push

# Open Prisma Studio (database GUI)
yarn prisma studio

# Seed database
yarn prisma db seed

# Reset database (warning: deletes all data)
yarn prisma db push --force-reset
```