User

September 4, 2026

User

Kind: Database Model

Source: `atloria-monorepo/libs/database/prisma/schema.prisma`

The `User` Prisma model represents an authenticated user within an organization. It stores identity, credentials, authorization role, organization membership, and optional GitHub/GitLab OAuth token data used for repository integrations.

Fields

Field	Type	Required	Key
`id`	`String`	✓	PK
`email`	`String`	✓	unique
`name`	`String`	✓	
`passwordHash`	`String`	-	
`role`	`UserRole`	✓	
`organizationId`	`String`	✓	
`githubAccessToken`	`String`	-	
`githubUsername`	`String`	-	
`gitlabAccessToken`	`String`	-	
`gitlabRefreshToken`	`String`	-	
`gitlabUsername`	`String`	-	
`bitbucketAccessToken`	`String`	-	
`bitbucketRefreshToken`	`String`	-	
`bitbucketUsername`	`String`	-	

Field	Type	Required	Key
`audienceIds`	`String[]`	✓	
`emailVerified`	`Boolean`	✓	
`emailVerificationToken`	`String`	-	unique
`passwordResetToken`	`String`	-	unique
`passwordResetExpiresAt`	`DateTime`	-	
`isActive`	`Boolean`	✓	
`scimExternalId`	`String`	-	
`provisionedBy`	`String`	-	
`isPlatformAdmin`	`Boolean`	✓	
`createdAt`	`DateTime`	✓	
`updatedAt`	`DateTime`	✓	
`suggestions`	`Suggestion[]`	✓	
`projectMembers`	`ProjectMember[]`	✓	
`createdTemplates`	`DocumentTemplate[]`	✓	
`collaborationSessions`	`CollaborationSession[]`	✓	
`pmeJobs`	`PmeGenerationJob[]`	✓	

Diagram

```
erDiagram
  USER {
    String id PK
    String email
    String name
    String passwordHash
    UserRole role
    String organizationId FK
    String githubAccessToken
    String githubUsername
    String gitlabAccessToken
    String gitlabRefreshToken
  }

  ORGANIZATION {
    String id PK
  }
```

```
  USER }o--|| ORGANIZATION : belongs_to
```

Usage

```
import { PrismaClient, UserRole } from '@prisma/client';

const prisma = new PrismaClient();

const user = await prisma.user.create({
  data: {
    id: crypto.randomUUID(),
    email: 'developer@example.com',
    name: 'Example Developer',
    passwordHash: await hashPassword('secure-password'),
    role: UserRole.MEMBER,
    organizationId: 'org_123',
    githubAccessToken: '',
    githubUsername: '',
    gitlabAccessToken: '',
    gitlabRefreshToken: '',
  },
});

const organizationUsers = await prisma.user.findMany({
  where: {
    organizationId: user.organizationId,
  },
  select: {
    id: true,
    email: true,
    name: true,
    role: true,
    githubUsername: true,
  },
});
```

AI Coding Instructions

- Always scope user lookups and mutations by `organizationId` when operating in an organization-specific context.
- Never return `passwordHash`, `githubAccessToken`, `gitlabAccessToken`, or `gitlabRefreshToken` from API responses; use Prisma `select` to expose safe fields only.
- Hash passwords before writing to `passwordHash`; do not store plaintext credentials.

- Use the `UserRole` enum for authorization checks instead of comparing arbitrary role strings.
- Treat GitHub and GitLab tokens as sensitive credentials, encrypt or securely manage them according to the application's token-storage conventions.

Relationships

- HAS_ONE → `Organization`
- HAS_ONE → `Document`
- HAS_ONE → `Document`
- HAS_ONE → `Comment`
- HAS_ONE → `Comment`
- HAS_ONE → `DocumentVersion`
- HAS_ONE → `DocumentRevision`
- HAS_ONE → `UserActivity`
- HAS_ONE → `ScreenshotJob`
- HAS_ONE → `Screenshot`
- HAS_ONE → `ScreenshotVersion`
- HAS_ONE → `ScreenshotComparison`
- HAS_ONE → `DocVersion`
- HAS_ONE → `DocIssueReport`
- HAS_ONE → `DocIssueComment`
- HAS_ONE → `DocIssueActivity`
- HAS_ONE → `Board`
- HAS_ONE → `BoardCard`

Referenced By

- `ProjectMember` (HAS_ONE)
- `Comment` (HAS_ONE)
- `Comment` (HAS_ONE)
- `Suggestion` (HAS_ONE)
- `DocumentTemplate` (HAS_ONE)
- `CollaborationSession` (HAS_ONE)
- `DocumentVersion` (HAS_ONE)

- `DocumentRevision` (HAS_ONE)
- `UserActivity` (HAS_ONE)
- `ScreenshotJob` (HAS_ONE)
- `Screenshot` (HAS_ONE)
- `ScreenshotVersion` (HAS_ONE)
- `ScreenshotComparison` (HAS_ONE)
- `PmeGenerationJob` (HAS_ONE)
- `DocIssueReport` (HAS_ONE)
- `DocIssueComment` (HAS_ONE)
- `DocIssueActivity` (HAS_ONE)
- `BoardCard` (HAS_ONE)