NepTube - Project Documentation 📺

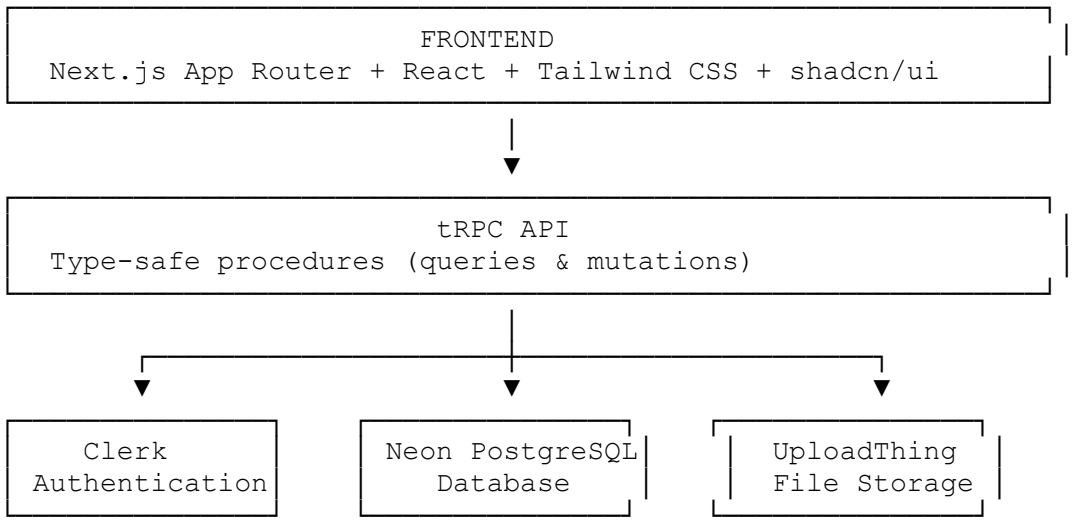
What is NepTube?
NepTube is a **YouTube-clone video sharing platform** built with modern web technologies. Users can upload, watch, search, and interact with videos, while admins can manage the platform.

🔗 GitHub Repository: https://github.com/Prabesh355/neptube

🛠️ Tech Stack

Technology	Purpose
Next.js 16	React framework with App Router, Server Components, Turbopack
TypeScript	Type-safe JavaScript
Clerk	Authentication (sign up, sign in, user management)
tRPC	Type-safe API calls between frontend and backend
Drizzle ORM	Database queries with TypeScript
Neon PostgreSQL	Serverless cloud database
UploadThing	File uploads (videos up to 512MB, thumbnails)
Pollinations AI	Free AI thumbnail generation
Tailwind CSS	Utility-first CSS styling
shadcn/ui	Pre-built UI components (buttons, cards, dialogs, etc.)
Bun	Fast JavaScript runtime & package manager

📐 Architecture



📁 Project Structure

```
src/
├── app/                        # Next.js App Router pages
│   ├── (auth)/               # Auth pages (sign-in, sign-up)
│   ├── (home)/               # Home layout
│   ├── admin/                # Admin panel (dashboard, users, videos,
settings)
│   ├── api/                  # API routes
│   │   ├── generate-thumbnail/  # AI thumbnail generation
│   │   ├── trpc/             # tRPC handler
│   │   ├── uploadthing/      # File upload handlers
│   │   └── users/webhook/    # Clerk webhook for user sync
│   ├── banned/               # Banned user page
│   ├── feed/                 # Video feed & video player
│   └── studio/               # Creator studio (upload, edit videos)
├── components/ui/            # shadcn/ui components
├── db/                       # Database schema & connection
├── lib/                      # Utilities
├── modules/                  # Feature modules (auth, home)
└── trpc/                     # tRPC router definitions
    └── routers/
        ├── admin.ts          # Admin procedures
        └── videos.ts         # Video procedures
```

✨ Features Implemented

👤 User Features
Feature	Description
Sign Up / Sign In	Email-based authentication via Clerk
Video Upload	Upload videos up to 512MB
AI Thumbnail Generation	Generate thumbnails using Pollinations AI
Manual Thumbnail Upload	Upload custom thumbnails
Search Videos	Search by title, description, or uploader name
Watch Videos	Video player with view count
Like/Dislike	React to videos
Creator Studio	Dashboard to manage your videos
Edit Videos	Change title, description, category, visibility,
thumbnail
Delete Videos	Remove your own videos

👑 Admin Features
Feature	Description
Admin Dashboard	Platform statistics (users, videos, views)
User Management	View all users, change roles, ban/unban
Video Management	View all videos, approve/reject/delete
Platform Settings	Configure site settings

🔒 Security Features
Feature	Description
Role-based Access	user, admin, moderator roles

Protected Routes	Admin panel only for admins
Ban System	Ban users from uploading/accessing
Middleware	Route protection via Clerk middleware

🗄️ Database Schema

Users Table
```sql
- id (UUID, primary key)
- clerk_id (text, unique) - Links to Clerk
- name (text)
- image_url (text)
- role (enum: user, admin, moderator)
- is_banned (boolean)
- ban_reason (text)
- created_at, updated_at
```

Videos Table
```sql
- id (UUID, primary key)
- user_id (UUID, foreign key)
- title (text)
- description (text)
- category (text)
- video_url (text)
- thumbnail_url (text)
- visibility (enum: public, private, unlisted)
- status (enum: draft, pending, published, rejected)
- view_count, like_count, dislike_count (integers)
- created_at, updated_at
```

Video Likes Table
```sql
- user_id (UUID)
- video_id (UUID)
- is_like (boolean) - true=like, false=dislike
```

🔑 Key Implementation Details

1. Authentication Flow
```
User clicks "Sign In" → Clerk modal opens → User enters email/password
→ Clerk authenticates → Middleware checks auth → User data synced to DB
```

2. Video Upload Flow
```
User goes to /studio/upload → Selects video file → UploadThing uploads to
cloud
→ User fills details (title, description) → Optionally generates AI
thumbnail
```

→ Video saved to database → Redirected to video page
```

### 3. AI Thumbnail Generation Flow
```
User enters title → Clicks "Generate AI Thumbnail"
→ API creates prompt → Pollinations AI generates image
→ Image downloaded → Uploaded to UploadThing → URL saved
```

### 4. Search Flow
```
User types in search bar → Presses Enter → URL updates with ?q=query
→ Feed page reads query → tRPC fetches videos matching
title/description/user
→ Results displayed
```

### 5. Admin Authorization
```
User accesses /admin → Middleware checks auth → tRPC checks user.role
→ If role !== 'admin' → Access denied
→ If admin → Full access to admin procedures
```

---

## 🌐 External Services

| Service | URL | Purpose |
|---|---|---|
| **Neon** | console.neon.tech | Database hosting |
| **Clerk** | dashboard.clerk.com | User authentication |
| **UploadThing** | uploadthing.com | File storage |
| **GitHub** | github.com/Prabesh355/neptube | Source code |

---

## 🚀 How to Run Locally

```bash
# Clone the repository
git clone https://github.com/Prabesh355/neptube.git
cd neptube

# Install dependencies
bun install

# Set up environment variables (create .env.local)
DATABASE_URL=your_neon_database_url
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=your_clerk_public_key
CLERK_SECRET_KEY=your_clerk_secret_key
UPLOADTHING_TOKEN=your_uploadthing_token

# Run development server
bun run dev

# Open in browser
```
```

http://localhost:3000
```

---

## 📸 Screenshots

### Home Feed
- Grid layout of video thumbnails
- Video title, uploader name, view count
- Search bar in navbar

### Video Player
- Full video player with controls
- Like/Dislike buttons
- Video description and uploader info

### Creator Studio
- Upload new videos
- Manage existing videos
- Edit video details
- AI thumbnail generation

### Admin Panel
- Dashboard with platform stats
- User management (ban/unban, roles)
- Video moderation (approve/reject/delete)

---

## 💡 Why These Technologies?

**Q: Why Next.js 16?**
A: Latest version with Turbopack for faster development, App Router for better routing, and Server Components for performance.

**Q: Why tRPC instead of REST API?**
A: tRPC provides end-to-end type safety - if you change an API, TypeScript catches errors immediately.

**Q: Why Clerk for authentication?**
A: Clerk handles all auth complexity (passwords, sessions, OAuth) with minimal code. Very secure and easy to implement.

**Q: Why Neon PostgreSQL?**
A: Serverless, scales automatically, generous free tier, works great with Drizzle ORM.

**Q: How does AI thumbnail generation work?**
A: We use Pollinations AI (free) which generates images from text prompts. The prompt is built from the video title/description.

---

## 👨‍💻 Developer

**Prabesh Basnet**
- GitHub: [@Prabesh355](https://github.com/Prabesh355)
```

📄 License

This project is for educational purposes.

Built with 💔 using Next.js, TypeScript, and modern web technologies