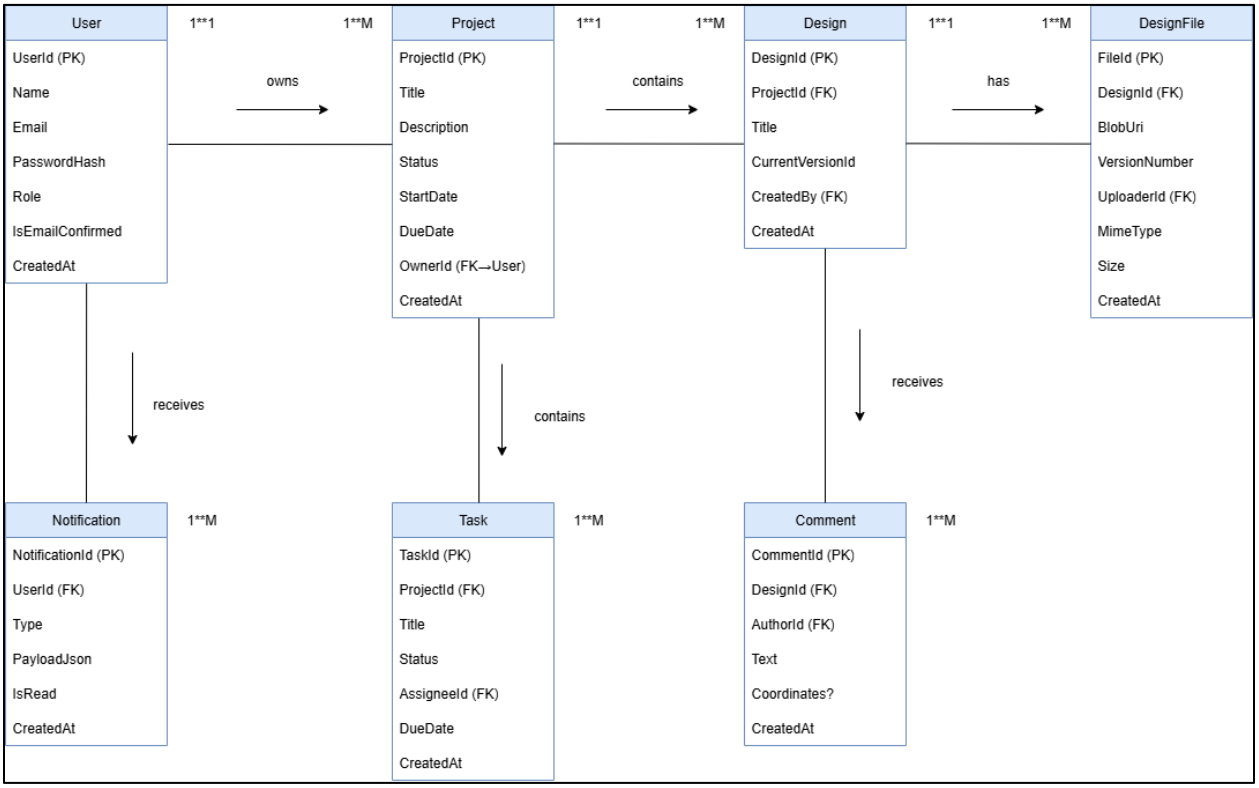
YanderFlow ERD:



The ERD for YanderFlow shows how the main parts of the application are structured and how they connect to each other. Each entity represents an important concept in the system, and the relationships explain how those concepts interact.

1. User

- Represents anyone using the system (Creative, Designer, or Client).
- Key information: name, email, role, account status.
- A **User can own projects, upload designs, leave comments, receive notifications, and be assigned tasks.**

2. Project

- Represents a design project or creative brief.
- Contains the title, description, deadlines, and overall status (e.g., Draft, In Review, Approved).

- A **User (Creative) owns the Project.**
- A **Project contains many Designs and Tasks.**

3. Design

- Represents a design created under a project.
- A Design belongs to a Project and is created by a User (Designer).
- A Design keeps track of its **current version** but can have many file versions.
- A **Project contains many Designs.**

4. DesignFile

- Represents the **uploaded design files** (e.g., images, PDFs).
- Each Design can have multiple versions stored as DesignFiles.
- Contains details like file location, version number, uploader, and size.
- A **Design has many DesignFiles (versions).**

5. Comment

- Represents feedback or annotations added to a design.
- Each Comment belongs to a specific Design and is authored by a User.
- This enables Clients and Creatives to give clear, actionable feedback.
- A **Design receives many Comments.**

6. Task

- Represents individual tasks within a project (e.g., “Upload first draft,” “Review design”).
- Each Task belongs to a Project and can be assigned to a specific User.
- Tasks have statuses (To Do, In Progress, Done) and due dates.
- A **Project contains many Tasks.**

7. Notification

- Represents updates sent to Users (e.g., new feedback, design approved, deadline reminders).
- Each Notification belongs to a User.
- Contains details like type of notification, extra data, whether it has been read, and timestamp.
- A **User receives many Notifications.**

8. Relationships

- A User owns Projects.
- A Project contains Designs and Tasks.
- A Design has many DesignFiles (versions).
- A Design receives Comments.
- A Task is assigned to a User.
- A User receives Notifications.

Outline System architecture:

Frontend (Client Layer)

- **Technology:** React + TypeScript
- **Responsibilities:**
  - Display project boards, briefs, designs, feedback, dashboards
  - Handle user interactions (uploading files, leaving comments, approving designs)
  - Communicate with backend via REST APIs and SignalR (for real-time updates)
- **Key Features:**
  - Role-based views (Creative, Designer, Client)
  - Responsive design for desktop + mobile
  - Drag-and-drop boards (task tracking, similar to Monday.com)

Backend (Application Layer)

- **Technology:** ASP.NET Core Web API
- **Responsibilities:**
  - Provide REST API endpoints for frontend communication
  - Enforce business rules (e.g., only Clients can approve designs)
  - Handle user authentication & authorization (ASP.NET Identity + JWT)
  - Process design uploads and manage version control
  - Manage notifications (real-time + email)
- **Key Modules:**
  - Authentication & Security
  - Projects & Briefs Management
  - Design Management (uploads, versions)
  - Feedback & Collaboration (comments, approvals)
  - Task Tracking (boards)
  - Notifications (SignalR + email)

Database (Data Layer)

- **Technology:** SQL Server (Entity Framework Core)
- **Responsibilities:**

    - Store structured application data (users, projects, designs, tasks, feedback)
    - Maintain relationships (e.g., Project → Designs → Files)
    - Ensure data integrity and version history
- **Entities:** User, Project, Design, DesignFile, Comment, Task, Notification

File Storage

- **Technology:** Azure Blob Storage
- **Responsibilities:**
    - Store uploaded design files (images, PDFs)
    - Handle multiple versions of files with secure URIs
    - Integrate with backend for access control

Notifications System

- **Real-Time (In-App):** SignalR hub sends live updates (e.g., new feedback, task status updates).
- **Email:** SendGrid (or SMTP) sends account-related emails (verification, password reset) and project updates.

Background Processing

- **Technology:** Hangfire or Azure Functions
- **Responsibilities:**
    - Send emails asynchronously
    - Generate file previews (optional)
    - Schedule reminders for deadlines

Security

- Role-based access control (Creative, Designer, Client)
- JWT authentication with refresh tokens
- Email/password login with password recovery via email link

- TLS (HTTPS) enforced across frontend and backend
- Input validation & file type/size restrictions

Hosting & Deployment

- **Frontend:** Azure Static Web Apps or CDN
- **Backend:** Azure App Service (with CI/CD pipeline)
- **Database:** Azure SQL Database
- **File Storage:** Azure Blob Storage
- **Monitoring:** Application Insights + Serilog logs