ERD & SYSTEM DESIGN

A. CORE ENTITIES (ERD DESIGN)

Your Productivity Suite has **four primary collections** and **two supporting concepts**.

1. User Entity

This is the **central entity**.

User

Field	Type	Description
_id	ObjectId	Primary key
name	String	User full name
email	String	Unique identifier
password	String	Hashed password
role	Enum	admin or standard
avatar	String	Profile image URL
isActive	Boolean	Account status
createdAt	Date	Timestamp
updatedAt	Date	Timestamp

Relationships

- One User → Many Notes
- One User → Many Tasks
- One User → Many Messages
- One User → Many Files

---

2. Notes Entity

Personal productivity notes.

Note

Field	Type	Description
_id	ObjectId	Primary key
title	String	Note title
content	String	Note body
tags	[String]	Optional categorization
owner	ObjectId (User)	Reference to User
isArchived	Boolean	Soft delete
createdAt	Date	Timestamp
updatedAt	Date	Timestamp

Relationship

- Many Notes → One User

---

3. Tasks Entity

User-specific task management.

Task

Field	Type	Description
_id	ObjectId	Primary key
title	String	Task title
description	String	Task details
status	Enum	pending, completed
dueDate	Date	Optional
assignedTo	ObjectId (User)	Task owner
createdBy	ObjectId (User)	Creator

Field	Type	Description
createdAt	Date	Timestamp
updatedAt	Date	Timestamp

Relationships

- Many Tasks → One User (assignedTo)
- Many Tasks → One User (createdBy)

---

4. Chat / Message Entity

Real-time messaging support.

Message

Field	Type	Description
_id	ObjectId	Primary key
sender	ObjectId (User)	Message sender
roomId	String	Chat room
content	String	Message body
isRead	Boolean	Read status
createdAt	Date	Timestamp

Relationship

- Many Messages → One User

---

5. File Entity (Optional but Recommended)

For profile images and attachments.

File

Field	Type	Description
_id	ObjectId	Primary key
fileUrl	String	Storage URL
fileType	String	image/pdf/etc
owner	ObjectId (User)	File owner
createdAt	Date	Timestamp

Relationship

- Many Files → One User

B. ERD RELATIONSHIP SUMMARY (ASSESSOR-FRIENDLY)

User 1 ——< Notes

User 1 ——< Tasks

User 1 ——< Messages

User 1 ——< Files

- All productivity data is **user-scoped**
- Admin users have **elevated access**, not separate tables
- Clean normalization
- Scales well in MongoDB

C. AUTHENTICATION FLOW DIAGRAM (SYSTEM DESIGN)

You will draw this as a **flow diagram**.

Authentication Flow

Client

↓

POST /api/auth/login

↓

Validate Credentials

↓

Generate JWT

↓

Return Token

↓

Client stores token

↓

Protected Route Request

↓

JWT Middleware

↓

Role Middleware

↓

Controller

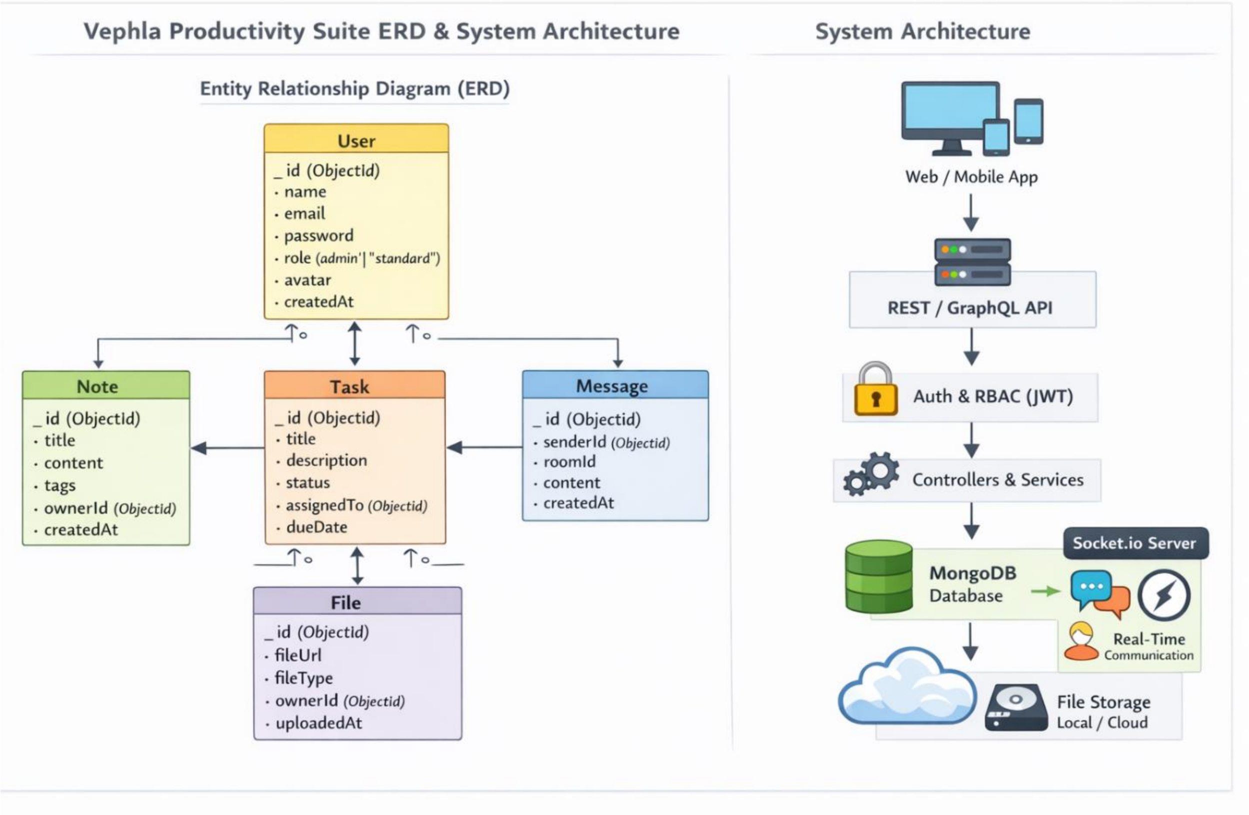
D. HIGH-LEVEL SYSTEM ARCHITECTURE DIAGRAM

This should be a **block diagram**.

```
Client (Web / Mobile)
     |
     v
  REST / GraphQL API
     |
     v
 Authentication & RBAC
```

```
     |
     v
 Controllers / Services
     |
     v
   MongoDB
     |
     v
  File Storage
(Cloudinary / Local)

+ Socket.io Server
  └── Real-time Events
```



Vephla Productivity Suite ERD & System Architecture

Entity Relationship Diagram (ERD)

User
- _ id (ObjectId)
- name
- email
- password
- role (admin'| "standard")
- avatar
- createdAt

Note
- _ id (ObjectId)
- title
- content
- tags
- ownerId (ObjectId)
- createdAt

Task
- _ id (ObjectId)
- title
- description
- status
- assignedTo (ObjectId)
- dueDate

Message
- _ id (Objectid)
- senderId (Objectid)
- roomId
- content
- createdAt

File
- _ id (ObjectId)
- fileUrl
- fileType
- ownerId (ObjectId)
- uploadedAt

System Architecture

Web / Mobile App

REST / GraphQL API

Auth & RBAC (JWT)

Controllers & Services

MongoDB Database

Socket.io Server

Real-Time Communication

File Storage Local / Cloud