devCollab: Social Media Platform (Final Version with SaaS V2 Plan)

Overview

devCollab is a MERN stack social media collaboration platform inspired by Reddit. Users can create accounts (via email/password, Google, or GitHub), post content, engage via comments/likes, follow other users, and chat privately or in groups (with video calls).

This project is designed as a scalable SaaS-ready product.

Final Features (Version 1)

Authentication

- Email/password registration and login
- Google Sign-In (OAuth2)
- GitHub Sign-In (OAuth2)

User System

- Separate User and UserProfile models
- Follow/follower logic
- Edit profile (bio, avatar, links, etc.)

Posts

- Create/update/delete posts
- Like posts

- Comment on posts
- Comment likes (upvotes)
- Realtime counters (likesCount, commentsCount)

Chat System

- One-to-one messaging
- Group chat (create/add/remove users)
- Message read receipts (`isRead`)
- Video calls (WebRTC-based)

UI Sections

- Feed page
- Profile page
- Post details page
- Chat interface (DMs & groups)
- Login/Register page

Final Data Models

1. User.js

```
const UserSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String },
  authProvider: { type: String, enum: ['local', 'google', 'github'], default: 'local' },
```

```
  googleId: { type: String },
  githubId: { type: String },
  profile: { type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' },
}, { timestamps: true });
```

2. UserProfile.js

```
const UserProfileSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  avatar: { type: String },
  bio: { type: String },
  socialLinks: {
    github: String,
    twitter: String,
    linkedin: String,
  },
  followers: [{ type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' }],
  following: [{ type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' }],
}, { timestamps: true });
```

3. Post.js

```
const PostSchema = new mongoose.Schema({
  content: { type: String, required: true },
  author: { type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' },
  likes: [{ type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' }],
```

```
  commentsCount: { type: Number, default: 0 },
  likesCount: { type: Number, default: 0 },
}, { timestamps: true });
```

4. Comment.js

```
const CommentSchema = new mongoose.Schema({
  content: { type: String, required: true },
  author: { type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' },
  post: { type: mongoose.Schema.Types.ObjectId, ref: 'Post' },
  likes: [{ type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' }],
}, { timestamps: true });
```

5. Message.js

```
const MessageSchema = new mongoose.Schema({
  sender: { type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' },
  receiver: { type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' },
  content: { type: String },
  isRead: { type: Boolean, default: false },
  chatRoom: { type: mongoose.Schema.Types.ObjectId, ref: 'ChatRoom' },
}, { timestamps: true });
```

6. ChatRoom.js

```
const ChatRoomSchema = new mongoose.Schema({
  name: { type: String },
```

```
  isGroup: { type: Boolean, default: false },
  participants: [{ type: mongoose.Schema.Types.ObjectId, ref: 'UserProfile' }],
}, { timestamps: true });
```

SaaS v2 Roadmap (Upcoming Version)

Future Features (v2)

- **Project Management**
    - Task lists, status tracking, contributors
    - Assign tasks to collaborators
- **Team Workspaces**
    - Invite-only team pages
    - Internal posts/comments within teams
- **Notifications (optional)**
    - Comment replies, mentions, follows
- **Analytics Dashboard**
    - Engagement stats, popular posts, user activity
- **Monetization**
    - Stripe/PayPal integration
    - Subscription plans (Pro accounts with more storage/rooms)

Development Stack

- **Frontend**: React.js + Tailwind CSS + Context/API
- **Backend**: Node.js + Express.js
- **Database**: MongoDB
- **Authentication**: Passport.js or Firebase Auth (for OAuth)
- **WebRTC**: Peer.js or custom socket signaling
- **Sockets**: Socket.IO for realtime messaging

Let me know if you'd like this converted into a downloadable PDF or want to begin implementation of a specific module (like authentication or posts).