File: shared/schema.ts

Type: ts | **Size:** 3648 bytes

```
import { pgTable, text, serial, integer, boolean, timestamp, jsonb } from "drizzle-orm/pg-core";
import { createInsertSchema } from "drizzle-zod";
import { z } from "zod";
export const users = pgTable("users", {
  id: serial("id").primaryKey(),
  username: text("username").notNull().unique(),
  password: text("password").notNull(),
  displayName: text("display_name"),
  email: text("email"),
  joinedAt: timestamp("joined_at").defaultNow(),
  intention: text("intention"),
  level: integer("level").default(1),
  levelProgress: integer("level_progress").default(0),
  // Subscription related fields
  isPremium: boolean("is_premium").default(false),
  subscriptionStatus: text("subscription_status").default("free"),
  subscriptionExpiry: timestamp("subscription_expiry"),
  stripeCustomerId: text("stripe_customer_id"),
  stripeSubscriptionId: text("stripe_subscription_id"),
});
export const insertUserSchema = createInsertSchema(users).pick({
  username: true,
  password: true,
  displayName: true,
  email: true,
  intention: true,
});
export const readings = pgTable("readings", {
  id: serial("id").primaryKey(),
  userId: integer("user_id"),
  question: text("question").notNull(),
  date: timestamp("date").defaultNow(),
  intention: text("intention"),
  cards: jsonb("cards").notNull(),
  summary: text("summary").notNull(),
  tags: text("tags").array(),
});
export const insertReadingSchema = createInsertSchema(readings).pick({
  userId: true,
  question: true,
  intention: true,
  cards: true,
  summary: true,
  tags: true,
});
export const dailyCards = pgTable("daily_cards", {
  id: serial("id").primaryKey(),
  userId: integer("user_id"),
  date: timestamp("date").defaultNow(),
  title: text("title").notNull(),
  message: text("message").notNull(),
  relationshipFocus: text("relationship_focus"),
  affirmation: text("affirmation"),
});
export const insertDailyCardSchema = createInsertSchema(dailyCards).pick({
  userId: true,
  title: true,
  message: true,
  relationshipFocus: true,
  affirmation: true,
});
// Types
export type User = typeof users.$inferSelect;
export type InsertUser = z.infer;
export type Reading = typeof readings.$inferSelect;
export type InsertReading = z.infer;
export type DailyCard = typeof dailyCards.$inferSelect;
export type InsertDailyCard = z.infer;
// Card types for the frontend
export type OracleCard = {
  id: string;
  title: string;
  position: string;
  icon: string;
  message: string;
  details: string;
  color: string;
};
export type ReadingResult = {
  question: string;
  intention: string;
  cards: OracleCard[];
  summary: string;
  date: Date;
  tags: string[];
};
export type DailyCardResult = {
  title: string;
  date: Date;
  message: string;
  relationshipFocus: string;
  affirmation: string;
  icon: string;
};
// Subscription related types
export type SubscriptionPlan = {
  id: string;
  name: string;
  description: string;
  features: string[];
  price: number;
  interval: 'month' | 'year';
  stripePriceId?: string;
};
export type SubscriptionStatus = 'free' | 'active' | 'canceled' | 'past_due' | 'incomplete';
export const premiumFeatures = [
  'Unlimited personalized readings',
  'Access to all deep analysis tools',
  'Advanced relationship compatibility tests',
  'Interactive "choose your own adventure" style readings',
  'Daily relationship insights and affirmations',
  'Ad-free experience',
  'Premium card designs',
  'Priority access to new features'
  // 'Love Circles group readings and discussions' - Feature hidden until we get more sign ups
];
```