猪猪银行 - 完整功能实现

现在让我们添加所有的交互功能，包括任务输入、星星计算、数据统计、徽章系统等。

步骤1：更新store（状态管理）

替换 `src/stores/useStore.ts` 的内容：

typescript

```typescript
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

interface Task {
  id: string;
  name: string;
  category: 'study' | 'exercise' | 'behavior' | 'creativity';
  completed: boolean;
  stars: number;
  date: string;
}

interface Achievement {
  id: string;
  name: string;
  description: string;
  icon: string;
  unlocked: boolean;
  unlockedDate?: string;
}

interface DailyRecord {
  date: string;
  tasks: Task[];
  totalStars: number;
  report?: string;
}

interface AppState {
  // 基础数据
  totalStars: number;
  currentStreak: number;
  dailyRecords: DailyRecord[];
  achievements: Achievement[];
  customTasks: Task[];

  // 操作方法
  addTask: (task: Omit<Task, 'id' | 'date'>) => void;
  completeTask: (taskId: string) => void;
  addCustomTask: (name: string, category: Task['category'], stars: number) => void;
  generateDailyReport: () => string;
  unlockAchievement: (achievementId: string) => void;
  getTodayTasks: () => Task[];
  getTodayProgress: () => number;
  getWeeklyStats: () => { totalStars: number; completionRate: number };
}
```

```
// 初始成就列表
const initialAchievements: Achievement[] = [
  {
    id: 'first-star',
    name: '初次闪耀',
    description: '获得第一颗星星',
    icon: '🌟',
    unlocked: false
  },
  {
    id: 'week-warrior',
    name: '周冠军',
    description: '连续7天完成任务',
    icon: '🏆',
    unlocked: false
  },
  {
    id: 'star-collector',
    name: '星星收集者',
    description: '累计获得100颗星星',
    icon: '💫',
    unlocked: false
  },
  {
    id: 'perfect-day',
    name: '完美一天',
    description: '单日获得所有星星',
    icon: '🎯',
    unlocked: false
  },
  {
    id: 'month-master',
    name: '月度大师',
    description: '连续30天完成任务',
    icon: '👑',
    unlocked: false
  }
];

// 默认任务模板
const defaultTaskTemplates = [
  { name: '完成作业', category: 'study' as const, stars: 3 },
  { name: '阅读30分钟', category: 'study' as const, stars: 2 },
  { name: '运动30分钟', category: 'exercise' as const, stars: 3 },
  { name: '帮助做家务', category: 'behavior' as const, stars: 2 },
  { name: '画画或手工', category: 'creativity' as const, stars: 2 }
```

```
];

export const useStore = create<AppState>()(
  persist(
    (set, get) => ({
      totalStars: 0,
      currentStreak: 0,
      dailyRecords: [],
      achievements: initialAchievements,
      customTasks: [],

      addTask: (taskData) => {
        const newTask: Task = {
          ...taskData,
          id: Date.now().toString(),
          date: new Date().toISOString().split('T')[0],
          completed: false
        };

        set((state) => {
          const today = new Date().toISOString().split('T')[0];
          const todayRecord = state.dailyRecords.find(r => r.date === today);

          if (todayRecord) {
            return {
              dailyRecords: state.dailyRecords.map(record =>
                record.date === today
                  ? { ...record, tasks: [...record.tasks, newTask] }
                  : record
              )
            };
          } else {
            return {
              dailyRecords: [...state.dailyRecords, {
                date: today,
                tasks: [newTask],
                totalStars: 0
              }]
            };
          }
        });
      },

      completeTask: (taskId) => {
        set((state) => {
          const today = new Date().toISOString().split('T')[0];
          let starsEarned = 0;
let totalTasksToday = 0;
let completedTasksToday = 0;

const newDailyRecords = state.dailyRecords.map(record => {
  if (record.date === today) {
    const newTasks = record.tasks.map(task => {
      if (task.id === taskId && !task.completed) {
        starsEarned = task.stars;
        return { ...task, completed: true };
      }
      return task;
    });

    totalTasksToday = newTasks.length;
    completedTasksToday = newTasks.filter(t => t.completed).length;

    return {
      ...record,
      tasks: newTasks,
      totalStars: newTasks.filter(t => t.completed).reduce((sum, t) => sum + t.stars, 0)
    };
  }
  return record;
});

const newTotalStars = state.totalStars + starsEarned;
const newAchievements = [...state.achievements];

// 检查成就解锁
if (newTotalStars >= 1 && !newAchievements.find(a => a.id === 'first-star')?.unlocked) {
  const achievementIndex = newAchievements.findIndex(a => a.id === 'first-star');
  newAchievements[achievementIndex] = {
    ...newAchievements[achievementIndex],
    unlocked: true,
    unlockedDate: new Date().toISOString()
  };
}

if (newTotalStars >= 100 && !newAchievements.find(a => a.id === 'star-collector')?.unlocked) {
  const achievementIndex = newAchievements.findIndex(a => a.id === 'star-collector');
  newAchievements[achievementIndex] = {
    ...newAchievements[achievementIndex],
    unlocked: true,
    unlockedDate: new Date().toISOString()
  };
}
```

```
if (completedTasksToday === totalTasksToday && totalTasksToday > 0 &&
    !newAchievements.find(a => a.id === 'perfect-day')?.unlocked) {
  const achievementIndex = newAchievements.findIndex(a => a.id === 'perfect-day');
  newAchievements[achievementIndex] = {
    ...newAchievements[achievementIndex],
    unlocked: true,
    unlockedDate: new Date().toISOString()
  };
}

// 计算连续天数
let newStreak = state.currentStreak;
const yesterday = new Date();
yesterday.setDate(yesterday.getDate() - 1);
const yesterdayStr = yesterday.toISOString().split('T')[0];

const yesterdayRecord = state.dailyRecords.find(r => r.date === yesterdayStr);
if (yesterdayRecord && yesterdayRecord.tasks.some(t => t.completed)) {
  newStreak = state.currentStreak + 1;
} else if (completedTasksToday > 0) {
  newStreak = 1;
}

// 连续天数成就
if (newStreak >= 7 && !newAchievements.find(a => a.id === 'week-warrior')?.unlocked) {
  const achievementIndex = newAchievements.findIndex(a => a.id === 'week-warrior');
  newAchievements[achievementIndex] = {
    ...newAchievements[achievementIndex],
    unlocked: true,
    unlockedDate: new Date().toISOString()
  };
}

if (newStreak >= 30 && !newAchievements.find(a => a.id === 'month-master')?.unlocked) {
  const achievementIndex = newAchievements.findIndex(a => a.id === 'month-master');
  newAchievements[achievementIndex] = {
    ...newAchievements[achievementIndex],
    unlocked: true,
    unlockedDate: new Date().toISOString()
  };
}

return {
  totalStars: newTotalStars,
  currentStreak: newStreak,
  dailyRecords: newDailyRecords,
  achievements: newAchievements
```

```
    };
  });
},

addCustomTask: (name, category, stars) => {
  const newTask = {
    name,
    category,
    stars
  };
  get().addTask(newTask);
},

generateDailyReport: () => {
  const today = new Date().toISOString().split('T')[0];
  const todayRecord = get().dailyRecords.find(r => r.date === today);

  if (!todayRecord) return '今天还没有任务记录哦！';

  const completedTasks = todayRecord.tasks.filter(t => t.completed);
  const totalTasks = todayRecord.tasks.length;
  const completionRate = totalTasks > 0 ? (completedTasks.length / totalTasks * 100).toFixed(0) : 0;

  const categoryStats = {
    study: completedTasks.filter(t => t.category === 'study').length,
    exercise: completedTasks.filter(t => t.category === 'exercise').length,
    behavior: completedTasks.filter(t => t.category === 'behavior').length,
    creativity: completedTasks.filter(t => t.category === 'creativity').length
  };

  let report = `📊 今日表现报告\n\n`;
  report += `完成率：${completionRate}% (${completedTasks.length}/${totalTasks})\n`;
  report += `获得星星：${todayRecord.totalStars}⭐\n\n`;
  report += `分类完成情况：\n`;
  report += `📚 学习：${categoryStats.study} 项\n`;
  report += `🏃 运动：${categoryStats.exercise} 项\n`;
  report += `😊 行为：${categoryStats.behavior} 项\n`;
  report += `🎨 创造：${categoryStats.creativity} 项\n\n`;

  if (Number(completionRate) === 100) {
    report += `🎉 太棒了！今天所有任务都完成了！`;
  } else if (Number(completionRate) >= 80) {
    report += `💪 很不错！继续加油！`;
  } else if (Number(completionRate) >= 60) {
    report += `😊 还可以，明天继续努力！`;
  } else {
    report += `💡 加油！明天会更好的！`;
    }

    return report;
  },

  unlockAchievement: (achievementId) => {
    set((state) => ({
      achievements: state.achievements.map(a =>
        a.id === achievementId
          ? { ...a, unlocked: true, unlockedDate: new Date().toISOString() }
          : a
      )
    }));
  },

  getTodayTasks: () => {
    const today = new Date().toISOString().split('T')[0];
    const todayRecord = get().dailyRecords.find(r => r.date === today);
    return todayRecord?.tasks || [];
  },

  getTodayProgress: () => {
    const todayTasks = get().getTodayTasks();
    if (todayTasks.length === 0) return 0;
    const completed = todayTasks.filter(t => t.completed).length;
    return (completed / todayTasks.length) * 100;
  },

  getWeeklyStats: () => {
    const today = new Date();
    const weekAgo = new Date();
    weekAgo.setDate(today.getDate() - 7);

    const weekRecords = get().dailyRecords.filter(record => {
      const recordDate = new Date(record.date);
      return recordDate >= weekAgo && recordDate <= today;
    });

    const totalStars = weekRecords.reduce((sum, record) => sum + record.totalStars, 0);
    const totalTasks = weekRecords.reduce((sum, record) => sum + record.tasks.length, 0);
    const completedTasks = weekRecords.reduce((sum, record) =>
      sum + record.tasks.filter(t => t.completed).length, 0);

    const completionRate = totalTasks > 0 ? (completedTasks / totalTasks) * 100 : 0;

    return { totalStars, completionRate };
  }
```

```
    }),
    {
      name: 'piggy-bank-storage',
    }
  )
);

// 初始化今天的默认任务
export const initializeTodayTasks = () => {
  const store = useStore.getState();
  const todayTasks = store.getTodayTasks();

  if (todayTasks.length === 0) {
    defaultTaskTemplates.forEach(template => {
      store.addTask(template);
    });
  }
};
```

步骤2：创建任务管理组件

创建 `src/components/TaskManager.tsx`：

typescript

```
import React, { useState } from 'react';
import { useStore } from '../stores/useStore';

export const TaskManager: React.FC = () => {
  const { addCustomTask } = useStore();
  const [showAddTask, setShowAddTask] = useState(false);
  const [taskName, setTaskName] = useState('');
  const [taskCategory, setTaskCategory] = useState<'study' | 'exercise' | 'behavior' | 'creativity'>('study');
  const [taskStars, setTaskStars] = useState(1);

  const handleAddTask = () => {
    if (taskName.trim()) {
      addCustomTask(taskName, taskCategory, taskStars);
      setTaskName('');
      setTaskStars(1);
      setShowAddTask(false);
    }
  };

  const categoryOptions = [
    { value: 'study', label: '📚 学习', color: 'bg-blue-100 text-blue-800' },
    { value: 'exercise', label: '🏃 运动', color: 'bg-green-100 text-green-800' },
    { value: 'behavior', label: '😊 行为', color: 'bg-yellow-100 text-yellow-800' },
    { value: 'creativity', label: '🎨 创造', color: 'bg-purple-100 text-purple-800' }
  ];

  return (
    <div className="bg-white rounded-2xl shadow-lg p-6">
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-2xl font-bold text-piggy-blue">任务管理</h2>
        <button
          onClick={() => setShowAddTask(!showAddTask)}
          className="bg-piggy-pink text-white px-4 py-2 rounded-lg hover:bg-pink-600 transition-colors"
        >
          + 添加任务
        </button>
      </div>

      {showAddTask && (
        <div className="mb-6 p-4 bg-piggy-cream rounded-lg">
          <div className="space-y-3">
            <input
              type="text"
              placeholder="任务名称"
              value={taskName}
              onChange={(e) => setTaskName(e.target.value)}
      className="w-full px-4 py-2 rounded-lg border border-gray-300 focus:outline-none focus:border-piggy
    />

    <div className="flex gap-2">
      {categoryOptions.map(option => (
        <button
          key={option.value}
          onClick={() => setTaskCategory(option.value as any)}
          className={`px-3 py-1 rounded-full text-sm font-medium transition-all ${
            taskCategory === option.value
              ? option.color
              : 'bg-gray-100 text-gray-600'
          }`}
        >
          {option.label}
        </button>
      ))}
    </div>

    <div className="flex items-center gap-3">
      <span className="text-gray-700">星星数：</span>
      <div className="flex gap-1">
        {[1, 2, 3, 4, 5].map(star => (
          <button
            key={star}
            onClick={() => setTaskStars(star)}
            className={`text-2xl ${star <= taskStars ? 'text-yellow-400' : 'text-gray-300'}`}
          >
            ⭐
          </button>
        ))}
      </div>
      <span className="text-piggy-orange font-bold">{taskStars}</span>
    </div>

    <div className="flex gap-2">
      <button
        onClick={handleAddTask}
        className="flex-1 bg-piggy-green text-white px-4 py-2 rounded-lg hover:bg-green-600 transition-colo
      >
        确认添加
      </button>
      <button
        onClick={() => {
          setShowAddTask(false);
          setTaskName('');
        }}
```

```
          className="flex-1 bg-gray-300 text-gray-700 px-4 py-2 rounded-lg hover:bg-gray-400 transition-colo
         >
          取消
         </button>
        </div>
       </div>
      </div>
     )}
    </div>
  );
};
```

步骤3：创建成就系统组件

创建 `src/components/AchievementDisplay.tsx`：

```typescript
import React from 'react';
import { useStore } from '../stores/useStore';

export const AchievementDisplay: React.FC = () => {
  const { achievements } = useStore();
  const unlockedCount = achievements.filter(a => a.unlocked).length;

  return (
    <div className="bg-white rounded-2xl shadow-lg p-6">
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-2xl font-bold text-piggy-blue">成就徽章</h2>
        <span className="text-piggy-orange font-bold">
          {unlockedCount}/{achievements.length} 已解锁
        </span>
      </div>

      <div className="grid grid-cols-3 gap-4">
        {achievements.map(achievement => (
          <div
            key={achievement.id}
            className={`text-center p-4 rounded-lg transition-all ${
              achievement.unlocked
                ? 'bg-yellow-100 scale-105 shadow-md'
                : 'bg-gray-100 opacity-50'
            }`}
          >
            <div className="text-4xl mb-2">{achievement.icon}</div>
            <h3 className="font-bold text-sm">{achievement.name}</h3>
            <p className="text-xs text-gray-600 mt-1">{achievement.description}</p>
            {achievement.unlocked && achievement.unlockedDate && (
              <p className="text-xs text-piggy-green mt-2">
                ✅ {new Date(achievement.unlockedDate).toLocaleDateString()}
              </p>
            )}
          </div>
        ))}
      </div>
    </div>
  );
};
```

步骤4：创建统计图表组件

创建 `src/components/StatsChart.tsx`：

typescript

```
import React from 'react';
import { useStore } from '../stores/useStore';

export const StatsChart: React.FC = () => {
  const { dailyRecords, currentStreak } = useStore();

  // 获取最近7天的数据
  const last7Days = Array.from({ length: 7 }, (_, i) => {
    const date = new Date();
    date.setDate(date.getDate() - (6 - i));
    return date.toISOString().split('T')[0];
  });

  const chartData = last7Days.map(date => {
    const record = dailyRecords.find(r => r.date === date);
    return {
      date: new Date(date).toLocaleDateString('zh-CN', { month: 'short', day: 'numeric' }),
      stars: record?.totalStars || 0
    };
  });

  const maxStars = Math.max(...chartData.map(d => d.stars), 10);

  return (
    <div className="bg-white rounded-2xl shadow-lg p-6">
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-2xl font-bold text-piggy-blue">本周统计</h2>
        <span className="text-piggy-orange font-bold">
          🔥 连续 {currentStreak} 天
        </span>
      </div>

      <div className="h-48 flex items-end gap-2">
        {chartData.map((data, index) => (
          <div key={index} className="flex-1 flex flex-col items-center">
            <div className="relative w-full">
              <div
                className="bg-gradient-to-t from-piggy-pink to-pink-300 rounded-t-lg transition-all duration-500 hov
                style={{
                  height: `${(data.stars / maxStars) * 160}px`,
                  minHeight: data.stars > 0 ? '20px' : '0px'
                }}
              >
                {data.stars > 0 && (
                  <div className="absolute -top-6 left-1/2 transform -translate-x-1/2 text-sm font-bold text-piggy-pink'
                    {data.stars}⭐
                  </div>
                )}
              </div>
            </div>
            <div className="mt-2 text-xs text-gray-600">{data.date}</div>
          </div>
        ))}
      </div>
    </div>
  );
};
```

步骤5：创建每日报告组件

创建 `src/components/DailyReport.tsx`：

typescript

```typescript
import React, { useState } from 'react';
import { useStore } from '../stores/useStore';

export const DailyReport: React.FC = () => {
  const { generateDailyReport } = useStore();
  const [showReport, setShowReport] = useState(false);
  const [report, setReport] = useState('');

  const handleGenerateReport = () => {
    const newReport = generateDailyReport();
    setReport(newReport);
    setShowReport(true);
  };

  return (
    <div className="bg-white rounded-2xl shadow-lg p-6">
      <h2 className="text-2xl font-bold text-piggy-blue mb-4">每日总结</h2>

      <button
        onClick={handleGenerateReport}
        className="w-full bg-gradient-to-r from-piggy-blue to-blue-500 text-white px-6 py-3 rounded-lg hover:op
      >
        生成今日报告 📊
      </button>

      {showReport && (
        <div className="mt-4 p-4 bg-blue-50 rounded-lg">
          <pre className="whitespace-pre-wrap text-sm text-gray-700">{report}</pre>
          <button
            onClick={() => setShowReport(false)}
            className="mt-3 text-sm text-piggy-blue hover:underline"
          >
            关闭
          </button>
        </div>
      )}
    </div>
  );
};
```

步骤6：更新主页面

替换 `src/pages/HomePage.tsx` 的内容：

typescript

```tsx
import React, { useEffect } from 'react';
import { useStore, initializeTodayTasks } from '../stores/useStore';
import { TaskManager } from '../components/TaskManager';
import { AchievementDisplay } from '../components/AchievementDisplay';
import { StatsChart } from '../components/StatsChart';
import { DailyReport } from '../components/DailyReport';

export const HomePage: React.FC = () => {
  const {
    totalStars,
    getTodayTasks,
    completeTask,
    getTodayProgress,
    getWeeklyStats
  } = useStore();

  const todayTasks = getTodayTasks();
  const todayProgress = getTodayProgress();
  const weeklyStats = getWeeklyStats();

  useEffect(() => {
    initializeTodayTasks();
  }, []);

  const handleCompleteTask = (taskId: string) => {
    completeTask(taskId);

    // 显示动画效果
    const button = document.getElementById(`task-${taskId}`);
    if (button) {
      button.classList.add('animate-bounce');
      setTimeout(() => {
        button.classList.remove('animate-bounce');
      }, 1000);
    }
  };

  const getCategoryColor = (category: string) => {
    switch (category) {
      case 'study': return 'bg-blue-100 text-blue-800';
      case 'exercise': return 'bg-green-100 text-green-800';
      case 'behavior': return 'bg-yellow-100 text-yellow-800';
      case 'creativity': return 'bg-purple-100 text-purple-800';
      default: return 'bg-gray-100 text-gray-800';
    }
  };
```

```
const getCategoryIcon = (category: string) => {
  switch (category) {
    case 'study': return '📚 ';
    case 'exercise': return '🏃 ';
    case 'behavior': return '😊 ';
    case 'creativity': return '🎨 ';
    default: return '⭐ ';
  }
};

return (
  <div className="min-h-screen bg-gradient-to-br from-pink-100 via-blue-50 to-yellow-50 p-4">
    <div className="max-w-6xl mx-auto">
      {/* 顶部标题栏 */}
      <header className="text-center py-8 mb-8">
        <h1 className="text-5xl font-bold text-piggy-pink mb-4 animate-pulse">
          🐷 猪猪银行 🐷
        </h1>
        <div className="flex justify-center items-center gap-6">
          <div className="bg-yellow-400 text-white px-8 py-4 rounded-full shadow-lg transform hover:scale-105 t
            <span className="text-3xl font-bold">⭐ {totalStars}</span>
            <span className="block text-sm">总星星</span>
          </div>
          <div className="bg-piggy-blue text-white px-6 py-3 rounded-full shadow-lg">
            <span className="text-xl font-bold">📈 {weeklyStats.completionRate.toFixed(0)}%</span>
            <span className="block text-sm">周完成率</span>
          </div>
        </div>
      </header>

      {/* 今日任务 */}
      <div className="grid grid-cols-1 lg:grid-cols-2 gap-6 mb-6">
        <div className="bg-white rounded-2xl shadow-lg p-6">
          <div className="flex justify-between items-center mb-4">
            <h2 className="text-2xl font-bold text-piggy-blue">今日任务</h2>
            <div className="text-sm text-gray-600">
              进度：{todayProgress.toFixed(0)}%
            </div>
          </div>

          {/* 进度条 */}
          <div className="w-full bg-gray-200 rounded-full h-3 mb-4">
            <div
              className="bg-gradient-to-r from-piggy-green to-green-500 h-3 rounded-full transition-all duration-5
              style={{ width: `${todayProgress}%` }}
            />
    </div>

    <div className="space-y-3">
     {todayTasks.map(task => (
      <div
       key={task.id}
       className={`flex items-center justify-between p-4 rounded-lg transition-all ${
        task.completed
         ? 'bg-green-50 opacity-75'
         : 'bg-piggy-cream hover:bg-yellow-100'
       }`}
      >
       <div className="flex items-center gap-3">
        <span className="text-2xl">{getCategoryIcon(task.category)}</span>
        <div>
         <span className={`font-medium ${task.completed ? 'line-through text-gray-500' : ''}`}>
          {task.name}
         </span>
         <div className="flex items-center gap-2 mt-1">
          <span className={`text-xs px-2 py-1 rounded-full ${getCategoryColor(task.category)}`}>
           {task.category === 'study' ? '学习' :
            task.category === 'exercise' ? '运动' :
            task.category === 'behavior' ? '行为' : '创造'}
          </span>
          <span className="text-xs text-gray-500">
           {task.stars} ⭐
          </span>
         </div>
        </div>
       </div>
       <button
        id={`task-${task.id}`}
        onClick={() => handleCompleteTask(task.id)}
        disabled={task.completed}
        className={`px-4 py-2 rounded-lg font-medium transition-all ${
         task.completed
          ? 'bg-gray-300 text-gray-500 cursor-not-allowed'
          : 'bg-piggy-green text-white hover:bg-green-600 transform hover:scale-105'
        }`}
       >
        {task.completed ? '已完成' : '完成'}
       </button>
      </div>
     ))}
    </div>
   </div>
```

```
          {/* 任务管理器 */}
          <TaskManager />
        </div>

        {/* 统计图表和每日报告 */}
        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6 mb-6">
          <StatsChart />
          <DailyReport />
        </div>

        {/* 成就系统 */}
        <AchievementDisplay />
      </div>
    </div>
  );
};
```

步骤7：添加动画效果

在 `src/index.css` 中添加自定义动画：

css

```css
@tailwind base;
@tailwind components;
@tailwind utilities;

/* 自定义动画 */
@keyframes bounce {
  0%, 100% {
    transform: translateY(0);
  }
  50% {
    transform: translateY(-10px);
  }
}

@keyframes pulse {
  0%, 100% {
    opacity: 1;
  }
  50% {
    opacity: 0.5;
  }
}

@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(10px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.animate-bounce {
  animation: bounce 1s ease-in-out;
}

.animate-pulse {
  animation: pulse 2s ease-in-out infinite;
}

.animate-fadeIn {
  animation: fadeIn 0.5s ease-out;
}
```

```css
/* 星星动画 */
@keyframes starPop {
  0% {
    transform: scale(0) rotate(0deg);
  }
  50% {
    transform: scale(1.2) rotate(180deg);
  }
  100% {
    transform: scale(1) rotate(360deg);
  }
}

.star-pop {
  animation: starPop 0.6s ease-out;
}

/* 成就解锁动画 */
@keyframes achievementUnlock {
  0% {
    transform: scale(0) rotate(-45deg);
    opacity: 0;
  }
  50% {
    transform: scale(1.1) rotate(10deg);
  }
  100% {
    transform: scale(1) rotate(0deg);
    opacity: 1;
  }
}

.achievement-unlock {
  animation: achievementUnlock 0.8s ease-out;
}

/* 进度条动画 */
@keyframes progressFill {
  from {
    width: 0%;
  }
}

.progress-fill {
  animation: progressFill 1s ease-out;
}
```

```
/* 卡片悬浮效果 */
.card-hover {
  transition: all 0.3s ease;
}

.card-hover:hover {
  transform: translateY(-5px);
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
}
```

步骤8：添加通知提醒功能

创建 `src/components/NotificationManager.tsx`：

typescript

```
import React, { useEffect } from 'react';

export const NotificationManager: React.FC = () => {
  useEffect(() => {
    // 请求通知权限
    if ('Notification' in window && Notification.permission === 'default') {
      Notification.requestPermission();
    }

    // 设置每日提醒
    const setupDailyReminders = () => {
      const now = new Date();

      // 早上9点提醒
      const morning = new Date(now);
      morning.setHours(9, 0, 0, 0);
      if (morning < now) {
        morning.setDate(morning.getDate() + 1);
      }

      // 下午3点提醒
      const afternoon = new Date(now);
      afternoon.setHours(15, 0, 0, 0);
      if (afternoon < now) {
        afternoon.setDate(afternoon.getDate() + 1);
      }

      // 晚上8点提醒
      const evening = new Date(now);
      evening.setHours(20, 0, 0, 0);
      if (evening < now) {
        evening.setDate(evening.getDate() + 1);
      }

      // 设置定时器
      const timers = [
        setTimeout(() => showNotification('早安！ ', '开始今天的任务吧 🌞 '), morning.getTime() - now.getTime()),
        setTimeout(() => showNotification('下午好！ ', '别忘了完成今天的任务哦 💪 '), afternoon.getTime() - now.getTi
        setTimeout(() => showNotification('晚上好！ ', '今天的任务都完成了吗？ 📝 '), evening.getTime() - now.getTime
      ];

      return () => timers.forEach(timer => clearTimeout(timer));
    };

    const showNotification = (title: string, body: string) => {
      if ('Notification' in window && Notification.permission === 'granted') {
        new Notification(title, {
          body,
          icon: '/icons/icon-192x192.png',
          badge: '/icons/icon-192x192.png',
          vibrate: [200, 100, 200]
        });
      }
    };

    setupDailyReminders();
  }, []);

  return null;
};
```

步骤9：更新App.tsx添加通知管理

更新`src/App.tsx`：

typescript

```typescript
import React, { useEffect } from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { HomePage } from './pages/HomePage';
import { NotificationManager } from './components/NotificationManager';

function App() {
  useEffect(() => {
    // 注册Service Worker
    if ('serviceWorker' in navigator) {
      window.addEventListener('load', () => {
        navigator.serviceWorker.register('/service-worker.js')
          .then(registration => console.log('SW registered: ', registration))
          .catch(registrationError => console.log('SW registration failed: ', registrationError));
      });
    }
  }, []);

  return (
    <Router>
      <NotificationManager />
      <Routes>
        <Route path="/" element={<HomePage />} />
      </Routes>
    </Router>
  );
}

export default App;
```

步骤10：添加PWA安装提示

创建`src/components/InstallPrompt.tsx`：

typescript

```
import React, { useState, useEffect } from 'react';

export const InstallPrompt: React.FC = () => {
  const [deferredPrompt, setDeferredPrompt] = useState<any>(null);
  const [showInstallPrompt, setShowInstallPrompt] = useState(false);

  useEffect(() => {
    const handler = (e: any) => {
      e.preventDefault();
      setDeferredPrompt(e);
      setShowInstallPrompt(true);
    };

    window.addEventListener('beforeinstallprompt', handler);

    return () => {
      window.removeEventListener('beforeinstallprompt', handler);
    };
  }, []);

  const handleInstallClick = async () => {
    if (!deferredPrompt) return;

    deferredPrompt.prompt();
    const { outcome } = await deferredPrompt.userChoice;

    if (outcome === 'accepted') {
      console.log('用户接受了安装');
    }

    setDeferredPrompt(null);
    setShowInstallPrompt(false);
  };

  if (!showInstallPrompt) return null;

  return (
    <div className="fixed bottom-4 left-4 right-4 md:left-auto md:right-4 md:w-96 bg-white rounded-lg shadow
      <div className="flex items-center gap-3">
        <div className="text-4xl">🐷 </div>
        <div className="flex-1">
          <h3 className="font-bold text-piggy-pink">安装猪猪银行</h3>
          <p className="text-sm text-gray-600">添加到主屏幕，随时记录成长！ </p>
        </div>
      </div>
      <div className="flex gap-2 mt-3">
```

```
          <button
            onClick={handleInstallClick}
            className="flex-1 bg-piggy-pink text-white px-4 py-2 rounded-lg hover:bg-pink-600"
          >
            安装
          </button>
          <button
            onClick={() => setShowInstallPrompt(false)}
            className="flex-1 bg-gray-200 text-gray-700 px-4 py-2 rounded-lg hover:bg-gray-300"
          >
            稍后
          </button>
        </div>
      </div>
    );
  };
```

将安装提示添加到HomePage：

typescript

```typescript
// 在 HomePage.tsx 的 return 语句最后添加
<InstallPrompt />
```

步骤11：修复依赖问题

如果遇到 `zustand/middleware` 的问题，更新安装命令：

bash

```bash
npm install zustand@latest
```

步骤12：创建猪猪图标

如果还没有图标，可以使用这个简单的方法创建：

1. 创建一个HTML文件 `create-icon.html`：

html

```
<!DOCTYPE html>
<html>
<head>
    <title>创建猪猪图标</title>
</head>
<body>
    <canvas id="canvas" width="512" height="512"></canvas>
    <script>
        const canvas = document.getElementById('canvas');
        const ctx = canvas.getContext('2d');

        // 背景
        ctx.fillStyle = '#FFE5B4';
        ctx.fillRect(0, 0, 512, 512);

        // 猪脸
        ctx.fillStyle = '#FFB6C1';
        ctx.beginPath();
        ctx.arc(256, 256, 150, 0, Math.PI * 2);
        ctx.fill();

        // 鼻子
        ctx.fillStyle = '#FF69B4';
        ctx.beginPath();
        ctx.ellipse(256, 280, 40, 30, 0, 0, Math.PI * 2);
        ctx.fill();

        // 鼻孔
        ctx.fillStyle = '#D1336F';
        ctx.beginPath();
        ctx.arc(240, 280, 8, 0, Math.PI * 2);
        ctx.fill();
        ctx.beginPath();
        ctx.arc(272, 280, 8, 0, Math.PI * 2);
        ctx.fill();

        // 眼睛
        ctx.fillStyle = '#000000';
        ctx.beginPath();
        ctx.arc(200, 220, 15, 0, Math.PI * 2);
        ctx.fill();
        ctx.beginPath();
        ctx.arc(312, 220, 15, 0, Math.PI * 2);
        ctx.fill();

        // 耳朵
        ctx.fillStyle = '#FFB6C1';
        ctx.beginPath();
        ctx.moveTo(150, 180);
        ctx.lineTo(120, 120);
        ctx.lineTo(180, 140);
        ctx.closePath();
        ctx.fill();

        ctx.beginPath();
        ctx.moveTo(362, 180);
        ctx.lineTo(392, 120);
        ctx.lineTo(332, 140);
        ctx.closePath();
        ctx.fill();

        // 下载链接
        const link = document.createElement('a');
        link.download = 'piggy-icon-512.png';
        link.href = canvas.toDataURL();
        link.click();
    </script>
</body>
</html>
```

2. 在浏览器中打开这个HTML文件，会自动下载一个512x512的猪猪图标
3. 使用在线工具 https://www.favicon-generator.org/ 生成其他尺寸

现在的功能包括：

1. ✅ **任务管理系统**
   - 默认每日任务
   - 自定义添加任务
   - 任务分类（学习、运动、行为、创造）
   - 任务完成标记
2. ✅ **星星积分系统**
   - 完成任务获得星星
   - 总星星统计
   - 每日/每周星星统计
3. ✅ **成就徽章系统**
   - 5个可解锁成就
   - 自动解锁机制

    - 解锁日期记录

4. ✅ **数据统计**
    - 7天星星趋势图
    - 连续完成天数
    - 周完成率统计
5. ✅ **每日报告**
    - 自动生成报告
    - 分类统计
    - 激励评语
6. ✅ **通知提醒**
    - 每日3次提醒
    - PWA安装提示
7. ✅ **数据持久化**
    - 本地存储所有数据
    - 刷新不丢失

重新部署到Vercel：

```bash
# 构建新版本
npm run build

# 部署
vercel --prod
```

现在您的应用应该有完整的交互功能了！