Recommended Project Structure

```
RomanceChatApp/
├── src/
│   ├── components/          # Reusable UI components
│   │   ├── common/
│   │   │   ├── Button/
│   │   │   │   ├── index.tsx
│   │   │   │   └── styles.ts
│   │   │   ├── Card/
│   │   │   │   ├── index.tsx
│   │   │   │   └── styles.ts
│   │   │   └── Avatar/
│   │   │       ├── index.tsx
│   │   │       └── styles.ts
│   │   ├── chat/
│   │   │   ├── MessageBubble/
│   │   │   │   ├── index.tsx
│   │   │   │   └── styles.ts
│   │   │   └── TypingIndicator/
│   │   │       ├── index.tsx
│   │   │       └── styles.ts
│   │   └── story/
│   │       ├── StoryCard/
│   │       │   ├── index.tsx
│   │       │   └── styles.ts
│   │       └── SceneChoice/
│   │           ├── index.tsx
│   │           └── styles.ts
│   │
│   ├── screens/             # Screen components
│   │   ├── StorySelectionScreen/
│   │   │   ├── index.tsx
│   │   │   └── styles.ts
│   │   ├── StorySceneScreen/
│   │   │   ├── index.tsx
│   │   │   └── styles.ts
│   │   └── ChatScreen/
│   │       ├── index.tsx
│   │       └── styles.ts
│   │
│   ├── navigation/          # Navigation configuration
│   │   ├── index.tsx
│   │   ├── types.ts
│   │   └── TabNavigator.tsx  # If you add tabs later
│   │
│   ├── services/            # API and external services
│   │   ├── api/
```

```
│   │   │   ├── client.ts
│   │   │   ├── chat.ts
│   │   │   └── types.ts
│   │   └── storage/
│   │       ├── sessionStorage.ts
│   │       └── userPreferences.ts
│   │
│   ├── hooks/                # Custom React hooks
│   │   ├── useChat.ts
│   │   ├── useStorage.ts
│   │   └── useStoryNavigation.ts
│   │
│   ├── utils/                # Utility functions
│   │   ├── errorHandling.ts
│   │   ├── constants.ts
│   │   ├── helpers.ts
│   │   └── validators.ts
│   │
│   ├── types/                # TypeScript type definitions
│   │   ├── index.ts
│   │   ├── story.ts
│   │   ├── chat.ts
│   │   └── navigation.ts
│   │
│   ├── styles/               # Styling and theme
│   │   ├── index.ts
│   │   ├── colors.ts
│   │   ├── typography.ts
│   │   ├── spacing.ts
│   │   └── components.ts
│   │
│   ├── assets/               # Static assets
│   │   ├── images/
│   │   │   ├── avatars/
│   │   │   ├── stories/
│   │   │   └── ui/
│   │   ├── fonts/
│   │   └── icons/
│   │
│   └── data/                 # Static data and content
│       ├── stories/
│       │   ├── index.ts
│       │   ├── christmasGift/
│       │   │   ├── index.ts
│       │   │   ├── scenes.ts
│       │   │   └── metadata.ts
│       │   ├── newBeginnings/
```

```
│   │   │   ├── index.ts
│   │   │   ├── scenes.ts
│   │   │   └── metadata.ts
│   │   └── anniversarySurprise/
│   │       ├── index.ts
│   │       ├── scenes.ts
│   │       └── metadata.ts
│   └── imageMap.ts
│
├── assets/                 # Expo assets (outside src)
│   ├── icon.png
│   ├── splash-icon.png
│   ├── adaptive-icon.png
│   └── favicon.png
│
├── App.tsx
├── index.ts
├── app.json
├── package.json
├── tsconfig.json
└── .gitignore
```

Key Improvements

1. **Component Organization**

- Separated reusable components by category (common, chat, story)
- Each component has its own folder with index.tsx and styles.ts
- Makes components more maintainable and testable

2. **Services Layer**

- Moved API logic to `services/api/`
- Separated storage logic into dedicated files
- Better separation of concerns

3. **Custom Hooks**

- Extract complex logic from components into reusable hooks
- Easier to test and maintain
- Better state management

4. **Type Definitions**

- Split types into logical files

- More organized and easier to maintain
- Better IntelliSense support

5. **Styles Organization**

- Separated theme constants into different files
- Component-specific styles with their components
- Centralized design system

6. **Story Data Structure**

- Each story in its own folder with multiple files
- Separated scenes from metadata
- More scalable for adding new stories

Migration Steps

1. **Create the new folder structure**
2. **Move existing files to appropriate locations**
3. **Update import paths throughout the project**
4. **Split large files into smaller, focused modules**
5. **Create reusable components**
6. **Extract custom hooks**
7. **Update TypeScript configurations**

Benefits

- **Scalability**: Easy to add new features and stories
- **Maintainability**: Clear separation of concerns
- **Reusability**: Components and hooks can be shared
- **Team Collaboration**: Clear structure for multiple developers
- **Testing**: Easier to write unit tests for individual components
- **Performance**: Better code splitting and lazy loading opportunities

This structure follows React Native and React best practices while being specific to your romance chat app's needs.