# SoulFun Project Documentation

Project Overview  
  
Full-stack SoulFun app: backend (CodeIgniter/PHP) and frontend (Next.js/React) delivering character, chat, image, and video features with a shared SoulCoin economy.  
Core services: character management, chat with OpenAI, image generation (google/imagen‑4 via Replicate), video generation (minimax/video‑01 via Replicate), and persistent galleries.  
  
Backend (CodeIgniter)  
- /app/Config/Routes.php routes requests onto API controllers under /api.  
- Controllers: Api\Characters, Api\Chats, Api\UserCharacters, Api\GeneratedImages, Api\GeneratedVideos, Api\Coins, Api\Auth.  
- Each returns JSON (ResponseTrait) and enforces session-based user\_id.  
- Libraries\CoinManager: wraps coin table access (user\_coins, user\_coin\_transactions), handles spend, refund, getBalance, and logs transactions for:  
 - Image generation (5 coins)  
 - Chat message (1 coin)  
 - Video generation (25 coins)  
 - Character creation (25 coins)  
  
Frontend (Next.js + React)  
- Layout: AppLayout with AppHeader (shows coins), AppSidebar, AppMobileNav.  
- Auth: AuthProvider provides user/session state and modal control.  
- Coins: CoinProvider fetches /coins, shares balance, refresh utility, and exposed useCoins.  
- Feature pages/components:  
 - components/generate/GenerateContent: prompt form, style/aspect controls, generates via /api/image-generator, saves to backend /generated-images, gallery, coin enforcement.  
 - components/video/VideoContent: text-to-video (active), other tabs “Coming soon”; generate via /api/video-generator, persist via /generated-videos, gallery management, coin enforcement.  
 - components/create/CreateContent: 4-step wizard to create/update characters; POST/PUT to backend, coin deducted for new entries.  
 - components/chats/ChatsContent: chat UI pulling history, sending messages to backend POST /chats/{slug}, updates coin balance.  
- API utilities: lib/api.ts centralizes fetch with fetchJson wrapper capturing HTTP status/errors. Exposes functions for characters, chats, images, videos, coins, and character CRUD.  
- Types: defined in lib/types.ts.  
  
API Integrations  
- Backend-proxied endpoints (frontend/app/api/backend/[[...path]]/route.ts) forward to PHP API with session cookies.  
- Image generation route (frontend/app/api/image-generator/route.ts):  
 - Requires REPLICATE\_API\_TOKEN.  
 - Calls google/imagen-4, polls until success (saves imageUrl).  
- Video generation route (frontend/app/api/video-generator/route.ts):  
 - Same token, minimax/video-01, polls up to 10 minutes, returns videoUrl.  
- These routes power frontend generate actions before saving to backend database.  
  
Persistence & Tables  
- generated\_images: user\_id, remote\_url, prompt, style, aspect\_ratio, quality, api\_type, timestamps.  
- generated\_videos: user\_id, remote\_url, prompt, model, duration\_seconds, thumbnail\_url, timestamps.  
- user\_coins: user\_id, balance, timestamps; user\_coin\_transactions: log of all adjustments.  
- user\_characters: stores custom characters (slug, name, personality, JSON traits, etc.).  
- ai\_messages: chat logs.  
  
Environment Variables  
Frontend/Next:  
- REPLICATE\_API\_TOKEN (required) for all generative routes.  
- NEXT\_PUBLIC\_API\_BASE\_URL (default /api/backend) and NEXT\_PUBLIC\_BACKEND\_BASE\_URL (defaults to local MAMP path) for API forwarding.  
- NEXT\_PUBLIC\_DEFAULT\_USER\_ID (optional fallback user id).  
  
Backend:  
- openai.apiKey, openai.model, openai.temperature, openai.endpoint for chat replies.  
- BACKEND\_API\_BASE\_URL used by frontend proxy (defaults to http://localhost:8888/soulsync-full/backend/public/api).  
- Ensure PHP session storage and database credentials configured in backend/config.  
  
Workflow Summary  
- User login handled by backend Api\Auth; frontend AuthProvider fetches /api/auth/me.  
- CoinProvider requests /api/coins after auth to display balance.  
- Generations: frontend posts to Next.js API (/api/image-generator or /api/video-generator) → Replicate → success returns URL → frontend saves via backend API → coin spent & record stored.  
- Character creation: wizard collects data → POST /api/backend/user-characters → backend spends coins → new character stored and balance returned.  
- Chats: send message POST /api/backend/chats/{slug} → backend expense & OpenAI reply → messages appended in UI.  
  
Common Error Handling  
- Coin deficit (HTTP 402): surfaces modal “You have no coins left.”  
- Replicate errors (e.g., insufficient billing, timeouts) bubble up via Next.js routes, displayed on cards (image/video).  
- Missing DB tables: backend failures respond with descriptive 500 error (e.g., “Video storage table is missing…”).  
  
Local Development Tips  
- Start backend (MAMP) exposing API at localhost:8888/soulsync-full/backend/public.  
- Run frontend npm run dev (Next 15). Restart after .env changes (especially REPLICATE\_API\_TOKEN) to reload serverless functions.  
- Seed user\_coins with desired balance (default 20). Refresh front-end to pick up manual changes.  
- Check Replicate billing status when receiving insufficient credit.  
  
Extending the System  
- Add new coin costs by updating CoinManager constants and lib/coins.ts.  
- For additional AI workflows, follow existing pattern: Next.js route → external API → backend persistence → update UI via useCoins and API helpers.  
- Ensure new backend endpoints are routed in Routes.php and use CoinManager as needed.  
  
This documentation should give developers the necessary map of functions, APIs, data flows, and configuration to work effectively with the project.