ChatGPT

Villages Community Reinvestment Platform – Frontend Specifications

Overview

This document provides a **comprehensive frontend handoff** for the Villages Community Reinvestment Platform. It builds on the underlying Move-based architecture and the UX flows described in the project's technical documentation. The goal is to provide a clear set of specifications and assets that a frontend engineer can use to implement a polished, accessible, and maintainable web application.

The platform's mission is to transform community participation (service hours and contributions) into **verifiable progress** on neighborhood projects. To achieve this, the frontend must connect to wallets (Petra or Aptos Connect), interact with Move modules on Aptos via the provided API endpoints, and present complex on-chain flows in a way that's easy for volunteers, staff, homeowners and external supporters to understand.

1. Visual & Interactive Assets

High-Fidelity Mockups

Although interactive prototypes are not provided in this document, the following list of **core screens** defines the scope for a high-fidelity design file (ideally created in Figma). These mockups should be designed to WCAG AA accessibility standards, follow the design system described below, and include all interactive states (empty, loading, populated, error, etc.).

1. **Landing / Home** – succinctly explains the purpose of the platform and invites users to connect a wallet. Use the handshake hero graphic below as a branding element. Provide calls to action for volunteers, homeowners, and supporters.
2. **Wallet Connect Modal** – allows users to choose Petra, Aptos Connect, or other supported wallets. Include installation links when `window.aptos` is not detected [1].
3. **Onboarding & KYC** – guides users through social login and identity verification. Present a consent screen explaining why KYC is required and how data is handled (off-chain).
4. **Volunteer Dashboard** – displays current Time Dollar balance, pending service requests, approved hours and redemption options. Contains flows to log service hours, view history, and stake Time Dollars into projects.
5. **Staff Dashboard** – for validators and admins. Shows pending hour approvals, member management, compliance controls, and treasury metrics.
6. **Project Catalog** – lists community projects (e.g., distressed homes) with progress bars, funding goals and impact share statistics. Users can view details, contribute, and monitor milestones.
7. **Project Detail** – deep dive into a single project: description, images, milestones, service and funding contributions, and a "stake/claim impact shares" call to action.

8. **Governance & Voting** – interface for proposals, voting and results. Include instructions for each voting mechanism (token-weighted, quadratic or conviction) [2].
9. **Settings & Account** – allow users to view their profile, connected wallets, roles, notifications preferences and KYC status.

Design files should include **user flow diagrams** that map the sequence of screens for key journeys (e.g., volunteer logs hours → pending approval → Time Dollars minted → redeem for impact shares). The user story map in the design sprint CSV provides a detailed breakdown of these flows and can be adapted into diagrams.

Assets

All icons and images should be exported as **SVG** (or PNG for raster) with meaningful names. The following assets are provided as part of this specification:

Asset	Description	Usage
	A stylized handshake within a circle, using deep purple and teal. This symbolises trust and collaboration.	Use on the landing page banner, headers or as a logo mark.
	A minimal icon combining a clock and a heart to represent Time Dollars (time-based credits earned through service).	Use next to Time Dollar balances, buttons to log hours, and tooltips explaining the currency.

To generate additional assets (e.g., project thumbnails or decorative motifs), follow similar visual motifs: simple shapes, smooth curves, and limited color palette.

2. Design System Specification

Color Palette

The palette balances the Village's community-oriented values with Aptos-inspired hues. Define these as CSS variables for easy theming.

Token	Usage	Hex
`--color-primary`	Buttons, highlights, icons	`#5E3FA3` (deep purple)
`--color-secondary`	Accents, links, callouts	`#00A0A6` (teal)
`--color-success`	Success states, approved hours	`#4CAF50`

Token	Usage	Hex
`--color-error`	Error states, invalid inputs	`#E53935`
`--color-warning`	Warnings, pending statuses	`#FFB300`
`--color-background`	Page background	`#F8F8F8` (off-white)
`--color-surface`	Cards, modals	`#FFFFFF`
`--color-on-primary`	Text on primary surfaces	`#FFFFFF`
`--color-text`	Main text	`#333333`
`--color-muted`	Secondary text, hints	`#777777`

Typography

Use **sans-serif** fonts for readability and approachability. Suggested stack:

- **Headings (H1–H3):** `Inter`, `Segoe UI`, `Helvetica Neue`, `Arial`, sans-serif. Bold for H1 (`32 px`), semi-bold for H2 (`24 px`), medium for H3 (`20 px`). Line heights of `1.25` to `1.3`.
- **Body:** `Inter`, `Roboto` or system UI at `16 px` with `24 px` line height. Use a lighter weight (`400`).
- **Caption/Help text:** `14 px` with `20 px` line height. Color should be `--color-muted`.

Spacing & Grid

Adopt an **8 px spacing system**. All margins, padding and component gaps should be multiples of 8 (`8 px`, `16 px`, `24 px`, `32 px`). Use a **12-column grid** for desktop layouts and a **4-column grid** for mobile.

Breakpoints

Define responsive breakpoints to ensure the UI adapts gracefully to various devices:

Name	Width	Description
`sm`	`@media (min-width: 480px)`	Small phones; single-column layout, stacked cards.
`md`	`@media (min-width: 768px)`	Tablets; 2–3 columns, bottom navigation replaced with side menu.
`lg`	`@media (min-width: 1024px)`	Small laptops; dashboard layouts with left sidebar, modals anchored.
`xl`	`@media (min-width: 1280px)`	Desktops; 12-column grid, multi-panel dashboards.

UI Components & States

Document each reusable component with its variations. For each, specify default, hover, focus, active, disabled, loading, and error states. Examples include:

- **Buttons** – Primary (solid background), Secondary (outlined), Tertiary (text). Add `aria-label` attributes for accessibility and show progress spinners when awaiting blockchain confirmations.
- **Inputs & Forms** – Text fields, number inputs, date pickers. Label everything clearly; use inline validation messages and icons for success/error. Provide accessible focus indicators.
- **Cards** – Used for projects, requests, user summaries. Elevate with subtle shadows (`box-shadow: 0 2 4 rgba(0,0,0,0.1)`) and 8 px radius. Cards can be clickable; indicate this on hover by slightly raising the card and darkening the shadow.
- **Navigation** – Side navigation for desktop, bottom tab bar on mobile. Icons accompanied by labels. Ensure enough contrast and larger touch targets (44 px). Highlight the current route with the primary color.
- **Modals & Drawers** – For flows like wallet connect, hour logging, redemption confirmation. Use overlay backgrounds with 50 % opacity; trap focus inside modals to comply with accessibility guidelines.
- **Table/List** components – Display lists of approvals, contributions, members. Include sorting, filtering and pagination controls. Support responsive stacking on smaller screens.

For each component, annotate any conditions or permissions (e.g., the “Approve” button in the staff dashboard should appear only for users with the validator role and the address must be whitelisted [3]).

Icons & Illustrations

- Use vector icons for clarity and scalability. Common actions such as wallet connection, KYC verification, approval, and redemption should have consistent iconography.
- The handshake and Time Dollar icons provided above serve as templates for the style: clean lines, limited color palette, and minimal details.
- Avoid using photographs of real people to ensure privacy and inclusivity. Instead, rely on simple illustrations or abstract patterns.

3. Technical Specifications

Aptos Network Endpoints

The frontend should be configurable for different Aptos networks. Use environment variables (`.env.local`) to store these URLs so that switching between Testnet and Mainnet is seamless during deployment.

Network	REST API	Indexer GraphQL API	gRPC	F
Mainnet	`https://api.mainnet.aptoslabs.com`	`https://api.mainnet.aptoslabs.com/v1/graphql`	`https://grpc.mainnet.aptoslabs.com`	–

Network	REST API	Indexer GraphQL API	gRPC	F
Testnet	`https://api.testnet.aptoslabs.com`	`https://api.testnet.aptoslabs.com/v1/graphql`	`https://grpc.testnet.aptoslabs.com`	U
Devnet	`https://api.devnet.aptoslabs.com`	`https://api.devnet.aptoslabs.com/v1/graphql`	`https://grpc.devnet.aptoslabs.com`	f

These endpoints are listed in the official Aptos documentation [4]. Use them to initialize the Aptos client in the frontend (e.g., with the `@aptos-labs/ts-sdk` or the wallet adapter). For testnet deployment, allow users to request test APT via the faucet.

Wallet Integration

Use the **Aptos Wallet Adapter** to support Petra and Aptos Connect. Wrap the application in `AptosWalletAdapterProvider` and access wallet functions via the `useWallet` hook. Implement these practices:

1. **Detection & onboarding** – Check for `window.aptos`. If not present, show a link to install Petra [1].
2. **Connect / Disconnect** – Use `wallet.connect()` to request connection and `wallet.disconnect()` to sign out [5].
3. **Transaction submission** – Call `signAndSubmitTransaction({ function: "0x…::module::method", typeArguments: [], arguments: [...] })` and display clear status messages (pending, succeeded, failed). In "gasless" mode, call your backend to request a sponsored transaction and display a spinner while it completes [6].
4. **Error handling** – Capture errors from the adapter and present human-readable messages; e.g., "Transaction rejected" if the user declines, or "Insufficient balance" if APT is needed for direct gas payments.

State Management & Data Fetching

- Use **React Query** or a similar library to manage asynchronous data from the REST API and indexer. Poll transaction status until it reaches `success`.
- Use GraphQL queries to fetch project lists, user balances, and event histories from the indexer. Cache these results and refetch on demand.
- For local development, provide fallback JSON files that mimic API responses. Replace them with real network calls once contracts are deployed on testnet.

Environment Configuration

Define the following environment variables (examples):

```
NEXT_PUBLIC_APTOS_NETWORK="testnet"
NEXT_PUBLIC_APTOS_REST_URL="https://api.testnet.aptoslabs.com"
```

```
NEXT_PUBLIC_APTOS_INDEXER_URL="https://api.testnet.aptoslabs.com/v1/graphql"
NEXT_PUBLIC_APTOS_GRPC_URL="https://grpc.testnet.aptoslabs.com"
NEXT_PUBLIC_APTOS_FAUCET_URL="https://aptosnetwork.github.io/aptos-core/wallet/
index.html"
```

Switch the values to Mainnet for production deployment.

Accessibility & UX Best Practices

- **Keyboard navigation:** Ensure every interactive element is reachable via keyboard. Provide focus outlines and ARIA labels. Trap focus within modals.
- **Loading & confirmation:** Always show progress indicators while waiting for blockchain confirmations. Provide success and error notifications with descriptions and actions (e.g., "View on Explorer").
- **Progressive disclosure:** Hide advanced features (e.g., gas fee inputs, governance options) by default. Offer an "advanced mode" toggle for power users.
- **Plain language:** Use simple explanations when dealing with blockchain concepts, such as "Add Time Dollars to Project" instead of "Stake tokens" [7].
- **Responsive design:** Test layouts on phones, tablets, and desktops. Provide larger touch targets and avoid horizontal scrolling.
- **Color contrast:** Use the color palette to meet or exceed WCAG AA contrast ratios. Never rely solely on color to convey meaning; pair colors with icons or text.

4. Component Library & States

Below is an overview of key components and their states. In the design file, create a **Component Library** page where each is documented with its variants.

Button

State	Appearance	Interaction
Default	Solid primary background with white text; 8 px radius	On hover, darken the background slightly; on focus, show an outline.
Disabled	Muted background and text; 40 % opacity; no shadow	Cursor shows `not-allowed` and clicking has no effect.
Loading	Replace label with spinner; maintain width	Display while awaiting transaction confirmation.
Success	For actions that change state, briefly change background to `--color-success` and show a checkmark.	

Input Field

State	Appearance	Notes
Default	Border: `1 px` solid `#CCCCCC`; placeholder in muted text	Show label above or inside the field using floating label pattern.
Focus	Border color changes to `--color-primary`; drop shadow added	Provide accessible focus indicator and ARIA description.
Error	Border and helper text in `--color-error`; error icon appended	Explain validation issue (e.g., “Hours must be a positive number”).
Disabled	Background fills with `#F5F5F5`; text muted	Do not allow input; used when user lacks the required role or KYC.

Card (Project / Request)

State	Description
Default	Shows title, short description, progress bar, and actions.
Hover	Card elevates and reveals additional actions (e.g., “View details”, “Contribute”).
Completed	Progress bar filled; show checkmark and label “Completed”.
Disabled	Gray out and display reason (e.g., “Project closed”).

Modal

Type	Use case
Confirm Transaction	Before calling `signAndSubmitTransaction`, display summary: function name (human-readable), cost (Time Dollars or gas), and user’s balances.
Success / Error	After transaction completes; show transaction hash and link to indexer or explorer.
Input Form	For logging service hours or submitting project details; embed the input fields described above.

Toast Notifications

Implement a toast system (bottom-right on desktop, top-center on mobile) for transient messages:

- **Info** – e.g., “Submitting your request…” with spinner.
- **Success** – e.g., “Hours approved. 5 Time Dollars minted!” with a green icon.
- **Error** – e.g., “Transaction failed: network error” with a red icon.
- **Warning** – e.g., “Relayer unavailable; switching to direct mode.”

5. Next Steps for Frontend Implementation

1. **Create a Figma file** using these specifications. Organize it into pages: *Components*, *Mockups*, *Flows*, *Style Guide*. Define components as reusable symbols with constraints.
2. **Build the React/Next.js project**. Install the `@aptos-labs/wallet-adapter-react` package and set up environment variables. Scaffold pages and components using a UI framework like Chakra UI or Tailwind CSS, customizing tokens to match the palette above.
3. **Integrate with the Move modules**. Use the provided TypeScript SDK or direct REST calls to interact with the `registry_hub`, `timebank`, `treasury`, `project_registry`, etc. Fetch data from the indexer GraphQL endpoint for project lists, event histories and balances. Use `view` functions on the Move modules for deterministic reads.
4. **Implement the relayer logic**. When the user is in gasless mode, forward transactions to the backend service which signs them and pays gas. Provide a fallback to direct transactions when the relayer is unavailable.
5. **Set up analytics and logging**. Track user actions (connect wallet, log hours, contribute, vote) anonymously to identify friction points. Respect privacy and regulatory requirements; do not store PII.
6. **Conduct usability testing**. Test with HCV staff and volunteers. Gather feedback on navigation clarity, terminology, and accessibility. Iterate based on findings before moving to mainnet.

Conclusion

The Villages Community Reinvestment Platform is more than a dApp—it is a civic tool designed to empower communities through transparent, on-chain participation. A thoughtful frontend design that blends modern web standards with blockchain-specific interactions is critical to its success. This specification provides the foundation for a frontend engineer to implement a **best-in-class, accessible user experience** that scales from initial pilots on testnet to broader deployments on mainnet.

---

1 5 Connecting to Petra Wallet – Petra
https://petra.app/docs/connect-to-petra

2 raw.githubusercontent.com
https://raw.githubusercontent.com/Homewood-Children-s-Village/Community-Reinvestment-Protocol/main/Technical%20Architecture.md

3 Community-Reinvestment-Protocol/villages_finance at main · Homewood-Children-s-Village/Community-Reinvestment-Protocol · GitHub
https://github.com/Homewood-Children-s-Village/Community-Reinvestment-Protocol/tree/main/villages_finance

4 Aptos Networks | Aptos Documentation
https://aptos.dev/network/nodes/networks

6 Time Dollar Sequence.png (1794×829)
https://raw.githubusercontent.com/Homewood-Children-s-Village/Community-Reinvestment-Protocol/main/Time%20Dollar%20Sequence.png

7 Designing for Blockchain: Try These 8 Best UX Practices
https://procreator.design/blog/designing-for-blockchain-best-ux-practices/