Suiciety – Frontend Page Specifications (MVP)

These specs describe **what each page does**, **what UI/UX elements exist**, **what data it needs**, and **which API endpoints are called**. They're written for quickly building an MVP in Replit with vanilla HTML/CSS/JS (or easily adaptable to React). Use them as a checklist while wiring to your backend.

Assumptions: - Roles: `CREATOR`, `SUPPORTER` (and optionally `ADMIN`). - Auth: JWT returned from `POST /auth/login` or `POST /auth/register`; token stored in `localStorage`. - Current endpoints (adapt as needed): - `POST /auth/register`, `POST /auth/login`, `POST /auth/logout` - `GET /users/me` (requires `Authorization: Bearer <token>`) - `PATCH /users/me` (update profile) - `GET /posts` (discover feed), `GET /posts/:id`, `POST /posts` (creator), `PATCH /posts/:id`, `DELETE /posts/:id` - `GET /studios` (list), `GET /studios/:id`, `POST /studios`, `PATCH /studios/:id` - `GET /studios/:id/members`, `POST /studios/:id/members` - `GET /creators/:id` (public creator page) - Set your API base URL in a single place: `const API = window.API_URL || 'https://api.example.com'` - All pages share a top nav that changes by auth state and role.

Global: Shared Layout & Utilities

Header / Navbar (all pages)

- **Logo / Home** → `index.html`
- **Discover** → `discover.html`
- **Studio** (if logged in & role=CREATOR) → `studio.html`
- **Help** → `help.html`
- **Search** input (global)
- **Auth area**:
- Logged out: `Login` (modal) · `Register` (link → `register.html`)
- Logged in: Avatar → menu: `Profile`, `Studio Data` (CREATOR), `Logout`
- **States**: loading (skeleton), authenticated (user mini-card), unauthenticated.

Footer (all pages)

- Links: Terms, Privacy, Contact, Language switch (optional)

Auth Utilities (JS)

- `getToken()`, `setToken(token)`, `clearToken()`
- `authFetch(url, options)` wrapper that injects `Authorization` header and handles 401 to redirect to `index.html` login modal.

UI States (Reusable)

- Loading spinners/skeletons
- Empty states (`No posts yet`, `No studios yet`)

- Error banners to surface API errors
- Toasts/snackbars for success/failure

---

1) `index.html` — Home / Landing

Goal: Explain the product quickly; show CTA to discover and sign up; show a minimal dynamic feed preview.

Sections

1. **Hero**
2. Headline, subheadline, `Get Started (Register)`, `Explore (Discover)`
3. If logged in → `Go to Studio` (CREATOR) or `Continue Exploring` (SUPPORTER)
4. **How it works** (3–4 cards): Create → Support → Grow → Withdraw (optional)
5. **Featured creators** (carousel of 6)
6. Data: `GET /creators?featured=true&limit=6`
7. **Trending posts** (preview of 6)
8. Data: `GET /posts?sort=trending&limit=6`
9. **Call-to-Action** – Join / Create your studio

Interactions

- Global search (redirect to `discover.html?q=<term>`)
- Click creator → `creator.html?id=<creatorId>`
- Click post → (optional detail page) or open post modal

Edge Cases

- API failures → show friendly fallback and a retry button

---

2) `register.html` — Register / Login

Goal: Provide entry to the platform with clear role selection.

Layout

- **Tabs**: `Login` | `Create Account`
- **Role picker** during registration: `CREATOR` or `SUPPORTER`
- Fields:
- **Register**: email, password, display name, role (radio), agree to terms
- **Login**: email, password
- Buttons: `Create account` / `Login`

API

- Register: `POST /auth/register` → `{ token, user }`
- Login: `POST /auth/login` → `{ token, user }`
- On success: store token, redirect:
- `CREATOR` → `studio.html`
- `SUPPORTER` → `discover.html`

Validation

- Basic: required, email format, password length
- Show server errors (email already used, wrong password) as inline messages

Extras

- "Continue as guest" → `discover.html` (limited features)
- "Forgot password?" (stub a modal or link to help page for now)

---

3) `profile.html` — User Profile (Private, editable)

Goal: Let a logged-in user manage their account info.

Sections

1. **Profile Header**
2. Avatar upload, Display name, Bio (textarea), Location (optional)
3. Save button (disabled if no changes)
4. **Account**
5. Email (read-only or editable with verify step), Change password (modal)
6. **Notifications**
7. Toggles: product updates, creator updates, supporter messages
8. **Role & Plan**
9. Show current role; if CREATOR, link to `studio.html`

Data Flow

- On load: `GET /users/me`
- Save: `PATCH /users/me` with changed fields
- Avatar: file upload (if backend supports) or stub with URL

Edge Cases

- 401 → redirect to `register.html`
- Optimistic UI with revert on failure

---

4) `studiodata.html` — Studio Data (Creator Analytics)

Goal: Provide creators a snapshot of performance.

Widgets (cards)

1. **Overview**: Total supporters, Total posts, Monthly earnings (if applicable)
2. API: `GET /studios/:id/metrics?range=30d`
3. **Supporters growth (chart)**
4. API: `GET /studios/:id/metrics/supporters?range=90d`
5. **Top posts** (table)
6. API: `GET /studios/:id/posts?sort=engagement&limit=10`
7. **Recent supporters** (list)
8. API: `GET /studios/:id/members?limit=10&sort=recent`

Filters

- Date range: 7d, 30d, 90d; updates all widgets

Access Control

- Only role=CREATOR and owner of the studio can access
- If no studio yet → CTA to create in `studio.html`

Empty States

- “No data yet. Post your first update to unlock analytics.”

---

5) `studio.html` — Creator Studio (Create & Manage)

Goal: Home for creators to publish and manage their content and membership.

Layout

- **Left nav**: Posts, Members, Settings
- **Main** (three views)

A. Posts

- New Post composer (title, content/markdown, media upload [stub ok])
- API: `POST /posts`
- Posts list (paginated)
- API: `GET /posts?owner=me`
- Actions per post: Edit, Delete, Publish/Unpublish
- `PATCH /posts/:id`, `DELETE /posts/:id`

B. Members

- Table: supporter name, joined date, contribution tier (if any)
- API: `GET /studios/:id/members`
- Invite link (copy button)
- Export CSV (client-side from current list)

C. Settings (for the studio)

- Studio name, description, cover image
- Public URL slug (read-only or editable with validation)
- Save: `PATCH /studios/:id`
- Danger zone: Delete studio (confirm twice)

Access Control

- Only role=CREATOR. If not authorized → show banner and link to `register.html` to switch role.

Edge Cases

- Draft vs Published states for posts
- Save guards on route change with unsaved edits

---

6) `help.html` — Help & Support

Goal: Provide FAQs and an easy way to get support.

Sections

1. **Search help** (client-side filter over FAQs)
2. **Popular topics**: Account, Payments (later), Posting, Discover
3. **FAQ accordion** (10–15 entries to start)
4. **Contact**: Simple form (email + message) — for MVP just open mailto or post to `/support/tickets` if available

Nice-to-have

- Link to status page (placeholder)
- “Report a bug” opens prefilled email subject

---

7) `discover.html` — Discover Feed (Public)

Goal: Let anyone browse creators and posts, with search & filters.

Controls

- Search bar → filters the list and updates URL param `?q=`
- Filter chips: Category (Art, Music, Writing…), Sort (Trending, Newest), Role (Creators only)

Content Areas

- **Creators grid** (avatar, name, short bio, follow/support button)
- API: `GET /creators?q=&category=&sort=&page=`
- **Posts feed** (optional for MVP; if included, ensure pagination)
- API: `GET /posts?q=&sort=&page=`

Interactions

- Click creator → `creator.html?id=<creatorId>`
- Support button: if logged-in supporter → follow action (stub); else open login/register

States

- Loading, Empty (no results), Error banner

---

8) `creator.html` — Public Creator Page

Goal: Public profile for a creator with a follow/support CTA.

Header

- Cover image, avatar, display name, handle/slug, bio, social links
- Buttons: `Support` (primary), `Follow` (secondary)

Body

- Tabs: `Posts` | `About`
- **Posts**: list of published posts by this creator
  - API: `GET /posts?creatorId=<id>&status=published`
- **About**: longer bio, studio info, join date, member count
  - API: `GET /creators/:id`

Sidebar (optional)

- Membership tiers (if used), perks list (static MVP acceptable)

Interactions

- `Support` button:
- If logged in as SUPPORTER → show confirmation modal (stubbed payment for MVP)
- If not logged in → open login/register

Minimal HTML/JS Skeleton (Reusable)

Use this base structure for each page and swap the `data-page` attribute to initialize page-specific logic in one `app.js` file.

```
<!doctype html>
<html lang="en" data-page="INDEX|REGISTER|PROFILE|STUDIODATA|STUDIO|HELP|
DISCOVER|CREATOR">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Suiciety</title>
  <link rel="stylesheet" href="styles.css" />
  <script>
    // Set your API URL once here or from env injection
    window.API_URL = "https://api.example.com";
  </script>
</head>
<body>
  <header id="nav"></header>
  <main id="app" class="container"></main>
  <footer id="footer"></footer>
  <script src="app.js"></script>
</body>
</html>
```

`app.js` outline

```
const API = window.API_URL;
const tokenKey = 'token';

function getToken(){ return localStorage.getItem(tokenKey); }
function setToken(t){ localStorage.setItem(tokenKey, t); }
function clearToken(){ localStorage.removeItem(tokenKey); }

async function authFetch(path, opts={}){
  const headers = Object.assign({ 'Content-Type': 'application/json' },
opts.headers||{});
  const token = getToken();
  if (token) headers['Authorization'] = `Bearer ${token}`;
  const res = await fetch(`${API}${path}`, { ...opts, headers });
  if (res.status === 401) { clearToken(); location.href = 'register.html';
return Promise.reject('Unauthorized'); }
  return res;
```

```
}

function initNav(user){ /* render nav based on user role */ }
async function getMe(){ try{ const r = await authFetch('/users/me'); return
r.ok ? r.json() : null; }catch{ return null; } }

async function boot(){
  const page = document.documentElement.dataset.page;
  const me = await getMe();
  initNav(me);
  switch(page){
    case 'INDEX': return renderIndex(me);
    case 'REGISTER': return renderRegister();
    case 'PROFILE': return renderProfile(me);
    case 'STUDIODATA': return renderStudioData(me);
    case 'STUDIO': return renderStudio(me);
    case 'HELP': return renderHelp();
    case 'DISCOVER': return renderDiscover();
    case 'CREATOR': return renderCreator();
  }
}

boot();
```

Tip: Keep each `renderX` small and split UI builders into separate modules when it grows.

Example: Page-specific DOM IDs & API calls

Below are quick DOM/ID sketches you can replicate per page.

`index.html`

- `#hero-cta-register` → link to `register.html`
- `#hero-cta-discover` → link to `discover.html`
- `#featured-creators` → cards filled from `/creators?featured=true`
- `#trending-posts` → cards from `/posts?sort=trending`

`register.html`

- `#tab-login`, `#tab-register`
- Login form: `#login-form` → POST `/auth/login`
- Register form: `#register-form` → POST `/auth/register`

`profile.html`

- `#profile-form` (displayName, bio, avatar)
- Save → `PATCH /users/me`

`studiodata.html`

- `#metrics-overview`, `#chart-supporters`, `#table-top-posts`, `#list-recent-supporters`
- Range selector: `#range-select` → refetch all widgets

`studio.html`

- Posts: `#post-editor`, `#posts-list`
- Members: `#members-table`, `#export-csv`
- Settings: `#studio-settings-form`

`help.html`

- Search: `#help-search`
- FAQ container: `#faq`
- Contact: `#contact-form` (mailto fallback)

`discover.html`

- Search input: `#discover-search`
- Filters: `.filter-chip`
- Creators grid: `#creators-grid`

`creator.html`

- Header: `#creator-header`
- Tabs: `#tab-posts`, `#tab-about`
- Support button: `#btn-support`

Accessibility & Performance

- Use semantic HTML (nav/main/section/article).
- Accessible labels for inputs and buttons.
- Keyboard focus states and skip links.
- Lazy-load images and paginate lists.

Next Steps Checklist (copyable)

- [ ] Implement shared `index.html` shell and `app.js` utilities.

- [ ] Build `register.html` first to unlock auth.
- [ ] Wire `profile.html` to `/users/me`.
- [ ] Create `studio.html` (Posts view) for publishing content.
- [ ] Publish `discover.html` listing creators.
- [ ] Add `studiodata.html` analytics with simple numbers (charts later).
- [ ] Fill `help.html` with static FAQ.
- [ ] Build `creator.html` to showcase public profile.

You can now drop these pages into Replit and iterate. Replace API paths as needed, and hook `renderX` functions to the DOM IDs specified above.