What “social media” actually includes

- **Core features (MVP):** sign up/sign in, profiles, posts (text/images), feed, likes/comments, basic search, file uploads.
- **Production features:** notifications (real-time), DMs, hashtags/topics, follows/blocks, reporting/moderation tools, rate limiting.
- **Serious stuff:** recommendation/ranking, spam/abuse detection, content moderation queues, accessibility, analytics, internationalization.
- **Infrastructure:** CDN + object storage for media, background jobs, caching, observability (logs/metrics/tracing), backups, CI/CD.
- **Compliance & safety:** privacy policy/ToS, age gates, DMCA, data retention, security reviews, GDPR/CPRA requests, incident response.

Path A (fastest, no backend to maintain): HTML + vanilla JS + Firebase (Auth, Firestore, Storage).

What to build (MVP)

- Sign up / login, profile (avatar, bio)
- Create post (text + ONE image)
- Follow/unfollow, reverse-chron home feed (you + people you follow)
- Like, comment
- Report post (simple admin “hide”) ←- I dont think we need it

Team roles (3–5 people)

- **Lead/PM (1):** backlog, scope, merges, demo. - Claudia

  Main goal - Keep everyone synced, ensure on-time delivery, handle merges, organize the final demo.

  Responsibilities

  - Define **MVP scope** (must-have vs nice-to-have).
  - Maintain the **Kanban/backlog** (To-Do, In Progress, Review, Done).
  - Run **daily 10-min standups** (who’s doing what, blockers).
  - Review PRs / merge clean code to `main`.
  - Manage **repo structure**, README, and project documentation.
  - Ensure a working **deployment link** by the end of Week 1.

- Coordinate **final presentation** slides/demo.

Deliverables

✅ Clean GitHub repo (organized folders, README, commit guidelines)
✅ Weekly progress reports
✅ Working deployed demo by Week 3
✅ Presentation slides (5 min walkthrough)

- **Frontend 1 (UI):** pages, forms, feed, accessibility. - Nino

  Main goal - Build and polish all visible pages and interactions.

  ### Responsibilities

  - Design layout (HTML + CSS) for:
    - `index.html` (login/signup)
    - `feed.html` (home feed)
    - `profile.html` (user profile)
    - `post.html` (single post view)
    - `admin.html` (report queue) ← dont need
  - Style all pages with a consistent theme (color palette, spacing, fonts).
  - Create responsive layout (desktop + mobile friendly).
  - Implement input forms, modals, navigation bar, buttons.
  - Handle **accessibility** (labels, alt text, focus order).
  - Add **loading states** and **error messages**.

  ### Deliverables

✅ Polished UI across all pages
✅ CSS file with reusable classes/components
✅ HTML templates wired to data placeholders (`<div id="feed">` etc.)

- **Frontend 2 (Data):** wiring to APIs (Firebase SDK or Express), state, pagination.- Edder

  Main goal: Connect the UI to backend/Firebase, handle state and dynamic rendering.

  ### Responsibilities

  - Write JS functions for:
    - Login/signup/logout.
    - Create/read posts from Firestore or API.
    - Follow/unfollow logic.
    - Likes/comments.
    - Feed pagination (load more).
  - Manage state in memory (`currentUser`, `feedPosts`, `followers`).
  - Integrate **Firebase SDK** or **API fetch calls** into UI.
  - Implement simple caching (avoid redundant reads).
  - Handle form validation and input sanitization.

  ### Deliverables

  ✅ `auth.js`, `posts.js`, `social.js` working modules
  ✅ Dynamic rendering of posts and profiles
  ✅ Real-time updates (optional: `onSnapshot` for Firebase)

- **DevOps/QA (rotating):** deploy, env secrets, test checklist, seed data.
  Main goal - Keep the project deployable, stable, and bug-free.

  Responsibilities

  - Deploy to **Firebase Hosting / Render / Vercel**.
  - Manage **env variables** (API keys, DB URI, storage bucket).
  - Write a test **checklist** (signup, post, follow, like, report).

- Conduct **manual testing** before merging PRs.
- Collect screenshots/GIFs for the final demo.
- Maintain **bug log** and verify fixes.

Deliverables

✅ Deployed working site
✅ Test checklist & QA report
✅ Backup/export of DB or Firestore data
✅ Demo screenshots

👥 5-Person Role Breakdown (No Backend Code)

Person	Role	Main Coding Work	Key Files / Areas	Weekly Focus
1. Project Lead / PM	Coordinates, merges, documents	Light JS edits, README, issue tracking	`README.md`, GitHub board, merge requests	Management + demo prep
2. Frontend UI Developer #1	Layout & styling	Builds all HTML + CSS structure (forms, feed cards, modals)	`index.html`, `feed.html`, `profile.html`, `admin.html`, `styles.css`	Design + responsiveness
3. Frontend UI Developer #2	Interactive elements	Adds DOM scripts for toggles, modals, navbars, and event listeners	`ui.js` or inline `<script>` sections	Client-side UX polish
4. Firebase Integrator	Connects to Firebase services	Initializes Firebase SDK, writes CRUD JS for posts/follows/likes	`firebase.js`, `auth.js`, `posts.js`, `social.js`	Data logic + Auth
5. DevOps / QA Engineer	Deployment + testing	Configures Firebase Hosting, writes test checklist	`firebase.json`, test_plan.docx`, screenshots	Deploy + quality control

1) Claudia

2) Nino

3) Edder

4) Azuka

5) Veronica

the **firebase integrator** is basically your **data + authentication developer** — the person who replaces what a traditional *backend developer* would normally do, but using Firebase's built-in services instead of writing server code.

🔧 In plain English

They make the front-end pages **talk to Firebase** — handling things like:

- signing up and logging in users
- saving posts, likes, and comments in Firestore
- uploading images to Firebase Storage
- writing security rules so users can't edit or delete each other's stuff
- connecting your HTML + JS buttons/forms to Firebase functions

---

💻 What they actually *code*

Mostly JavaScript that connects to Firebase's SDK:

Area	Example code they write
Firebase setup	Initialize SDK in `firebase.js` and export handles for auth, database, and storage.
Authentication	Functions in `auth.js` for signup, login, logout.
Posts	Functions in `posts.js` to add/read/delete posts from Firestore.
Likes/Comments	Functions in `social.js` or `interact.js` to update subcollections.
Image uploads	Code using `firebase.storage()` to upload and get image URLs.
Security	Set up Firestore/Storage rules in the Firebase console or `firestore.rules`.

Delegation & timeline

Week 1 – Foundations & first deploy

- *Frontend 1*: skeleton pages, navbar, forms; styles.
- *Frontend 2*: Firebase init, auth flows, profile CRUD.
- *QA*: seed script (create 5 users, 50 posts).
- *Lead*: Hosting setup, env secrets, DoD + PR template.
  Deliverable: Live site with auth + create post + basic feed (your posts only).

Week 2 – Social & interactions

- *Frontend 2*: follow/unfollow; query feed = self + followees (order by `createdAt desc`, paginate by `startAfter`).
- *Frontend 1*: post card UI; like/comment UI (optimistic).
- *QA*: accessibility pass (labels, alt text), smoke tests.
- *Lead*: Firestore rules draft (read/write by auth, size/type checks).
  Deliverable: working home feed, likes/comments, profiles.

Week 3 – Moderation, polish, demo

- *Frontend 2*: report flow; admin list + hide (flip `hidden:true`).
- *Frontend 1*: skeleton loaders, empty states, error toasts.
- *QA*: bug bash; demo script; screenshots.
- *Lead*: finalize rules, backups (export), README.
  Deliverable: polished MVP + 5-min demo.