MVP Checklist (Must Complete First)

Monorepo Setup

- Create root project folder lattice/
- Inside lattice/, create client/ and server/
- Create root package.json
- Install concurrently at root:

  ```
  npm install -D concurrently
  ```

- Configure root package.json scripts:

  ```
  {
    "scripts": {
      "install-all": "npm --prefix server install && npm --prefix client install",
      "dev": "concurrently \"npm run server\" \"npm run client\"",
      "server": "npm --prefix server run dev",
      "client": "npm --prefix client run dev"
    }
  }
  ```

Backend (/server)

Setup:

- Initialize package.json in /server
- Install dependencies:

  npm install express mongoose
  npm install -D nodemon

- Create folder structure:

  ```
  /config/connection.js
  /controllers/
  /models/
  /routes/
  /utils/ (optional)
  server.js
  ```

- Database:
- Models:
  - /models/User.js
  - /models/Thought.js (embed Reaction schema inside)
  - Add virtuals:
    - User: friendCount
    - Thought: reactionCount
  - Add timestamp formatting (getter)
- Controllers:
  - /controllers/userController.js
    - `CRUD` for User
    - Add/remove Friend
  - /controllers/thoughtController.js
    - `CRUD` for Thought
    - Add/remove Reaction
- Routes:
  - /routes/api/userRoutes.js
  - /routes/api/thoughtRoutes.js
  - /routes/index.js (combine routes)
  - Mount /api routes in server.js
- Server:
  - Setup express() server
  - Use express.json() middleware
  - Listen on `PORT 3001`
- Testing:
- Test all API routes with **Insomnia**:
  - `GET` all/single users
  - `POST/PUT/DELETE` users
  - `POST/DELETE` friends
  - `GET` all/single thoughts
  - `POST/PUT/DELETE` thoughts
  - `POST/DELETE` reactions

Frontend (/client)

Setup:

- Initialize package.json in /client
- Install Vite + React:

  ```
  npm install vite react react-dom
  npm install -D @vitejs/plugin-react
  ```

- Create folder structure:

  ```
  /config/connection.js
  /controllers/
  /models/
  /routes/
  /utils/ (optional)
  server.js
  ```

- Create Vite config vite.config.js with proxy:

  ```
  export default defineConfig({
    plugins: [react()],
    server: {
      port: 5173,
      proxy: {
        '/api': 'http://localhost:3001'
      }
    }
  });
  ```

- Structure:
  - /public/index.html (basic)
  - /src/main.jsx
  - /src/App.jsx
- Basic Pages:
  - Homepage (feed of thoughts)
  - Login/Signup (simple form for username + email)
  - User Dashboard:
    - View friends list
    - View own posts
  - Add Friend functionality
  - Create New Thought
  - React to Thought
  - Comment on Thought (can use reaction for now if needed)
- API Interaction:
  - Create simple /src/utils/api.js for Axios calls (or fetch)
  - Call backend routes: login, create thought, list friends, post reactions
- MVP UI Goal:
  - Simple, functional
  - Enough for you to **demo the full flow**: create user ➔ post thought ➔ react ➔ add friend ➔ view dashboard

Documentation

- Write a high-quality README:
    - What it is
    - How to run it
    - Link to video walkthrough
- Push commits often with descriptive messages

Stretch Goals (if you have time after MVP)

Networks (Mini-Subreddits)

- Create `Network` model:
    - Name
    - Description
    - Posts linked to network

- Allow users to join a Network
- Create Network pages (view network posts)

Image Uploads

- Allow posts with optional images (basic file upload)
- Store image URLs in database (or in local / cloud for now)

Notifications

- Notify when friend posts a new Thought

Profile Enhancements

- Profile bios
- Profile pictures (optional)

UI/UX Improvements

- TailwindCSS or simple custom CSS
- Nicer form validation
- Loading spinners, empty state displays

Deployment

- Configure monorepo deployment on **Render**:
    - Set up a **new Render Web Service** (for the monorepo root).
- Use **Render build & start commands**:
    - **Build Command**:

      ```
      npm run install-all && npm run build
      ```

    - **Start Command:**

      ```
      npm run dev
      ```

- Make sure client Vite proxy is set correctly (already done if `/api` points to backend).
- Add environment variables on Render if needed (e.g., database URI).
- Ensure MongoDB Atlas or another production database is used (unless you self-host MongoDB somewhere).