Social Network API — Backend Project Plan

Setup

1. **Initialize the project**
    a. Run `npm init -y` to create a `package.json`.
    b. Install dependencies:
    c. npm install express mongoose
    d. (Optional) Install a date formatting library if desired (or just use native JavaScript `Date`).
2. **Create the project folder structure**
    a. /config/ → Database connection
    b. /models/ → Mongoose models (User, Thought, Reaction schema)
    c. /controllers/ → Logic for API routes
    d. /routes/ → Express routes
    e. /utils/ → (Optional) Format timestamp utilities
    f. server.js → Entry point
3. **Connect to MongoDB**
    a. Create a `config/connection.js` file.
    b. Use Mongoose to connect to a local MongoDB instance: `mongodb://127.0.0.1:27017/socialNetworkDB`.

Models

4. **Create Mongoose models**
    a. User Model (`/models/User.js`)
        i. Fields:
            1. `username`: String, required, unique, trimmed
            2. `email`: String, required, unique, must match valid email
            3. `thoughts`: [Array of _id references to Thought model]
            4. `friends`: [Array of _id references to User model (self-reference)]
        ii. Schema Settings:
            1. Virtual: `friendCount` → number of friends
    b. Thought Model (`/models/Thought.js`)
        i. Fields:
            1. `thoughtText`: String, required, 1–280 characters
            2. `createdAt`: Date, default now, getter to format timestamp
            3. `username`: String, required (user who created)
            4. `reactions`: [Array of Reaction subdocuments]
        ii. Schema Settings:
            1. Virtual: `reactionCount` → number of reactions
    c. Reaction Schema (embedded in Thought, **not a separate model**)
        i. Fields:
            1. `reactionId: ObjectId, default new ObjectId`
            2. `reactionBody: String, required, max 280 characters`
            3. `username: String, required`
            4. `createdAt: Date, default now, getter to format timestamp`

API Routes

5. **Build User Routes (`/routes/api/userRoutes.js`)**
   a. `GET /api/users` → Get all users
   b. `GET /api/users/:userId` → Get one user (populate thoughts and friends)
   c. `POST /api/users` → Create user
   d. `PUT /api/users/:userId` → Update user
   e. `DELETE /api/users/:userId` → Delete user **(BONUS: also delete user's thoughts)**
   f. `POST /api/users/:userId/friends/:friendId` → Add friend
   g. `DELETE /api/users/:userId/friends/:friendId` → Remove friend
6. **Build Thought Routes (`/routes/api/thoughtRoutes.js`)**
   a. `GET /api/thoughts` → Get all thoughts
   b. `GET /api/thoughts/:thoughtId` → Get one thought
   c. `POST /api/thoughts` → Create thought (and push thought to user's thoughts array)
   d. `PUT /api/thoughts/:thoughtId` → Update thought
   e. `DELETE /api/thoughts/:thoughtId` → Delete thought
7. **Build Reaction Routes (Nested under Thought routes)**
   a. `POST /api/thoughts/:thoughtId/reactions` → Add reaction to thought
   b. `DELETE /api/thoughts/:thoughtId/reactions/:reactionId` → Remove reaction from thought

Controllers

8. **Create controller files (`/controllers/userController.js`, `/controllers/thoughtController.js`)**
   a. Each controller should handle logic for each route:
      i. Create, read, update, delete users
      ii. Create, read, update, delete thoughts
      iii. Add/remove friends
      iv. Add/remove reactions

Server

9. **Set up server.js**
   a. Import Express
   b. Set middleware for JSON parsing
   c. Mount routes under `/api`
   d. Listen on a port (e.g., 3001)

```
const PORT = process.env.PORT || 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Testing

10. **Use Insomnia to test API routes**
    a. Test GET all users and thoughts
    b. Test GET single user and thought
    c. Test POST, PUT, DELETE for users and thoughts
    d. Test POST, DELETE for friends
    e. Test POST, DELETE for reactions
11. **Record a video walkthrough**

a. Show starting the server
b. Demonstrate all required API routes working in Insomnia:
    i. GET all/single users
    ii. GET all/single thoughts
    iii. POST, PUT, DELETE users
    iv. POST, PUT, DELETE thoughts
    v. POST, DELETE friends
    vi. POST, DELETE reactions
c. Submit video link in your README file

Deliverables Checklist

- GitHub repo with full project code
- High-quality README with project description and video link
- Walkthrough video demonstrating functionality
- Proper commits with meaningful messages

🔥 Bonus for Extra Credit

When deleting a user, **also delete** all associated thoughts automatically.